مدينة الملك عبد العزيز للعلوم والتقنية

المنظمة العربية للترجمة

دليل تطوير البرمجيّات الشامل



راجفيندر سانغوان - ماثيو باس - نيل موليك
دانيال ج. باوليش - جيورغين كازمير

ترجمة

مرفت سلمان

سلسلة كتب التقنيات الاستراتيجية والمتقدّمة

دلـــيـــل تطوير
البرمجيّات الشامل

المنظمة العربية للترجمة

راجفيندر سانغوان ـ ماثيو باس ـ نيل موليك
دانيال ج. باوليش ـ جيورغين كازميير

دليل تطوير البرمجيّات الشامل

ترجمة

مرفت سلمان

مراجعة

أسامة فرّوخ

توزيع: **مركز دراسات الوحدة العربية**

الفهرسة أثناء النشر ـ إعداد المنظمة العربية للترجمة
دليل تطوير البرمجيّات الشامل/ راجفيندر سانغوان [وآخرون]؛ ترجمة مرفت سلمان؛ مراجعة أسامة فرّوخ.
428 ص. ـ (تقنيات استراتيجية ومتقدمة ـ تقنية المعلومات؛ 1)
يشتمل على فهرس.
ISBN 978-9953-0-1994-9
1. البرمجة (حاسبات إلكترونية). 2. هندسة النظم. أ. سانغوان، راجفيندر. ب. سلمان، مرفت (مترجم). ج. فرّوخ، أسامة (مراجع). د. السلسلة.
005.1

«الآراء الواردة في هذا الكتاب لا تعبّر بالضرورة
عن اتجاهات تتبناها المنظمة العربية للترجمة»

Global Software Development Handbook
Edited by Raghvinder Sangwan [et al.].

المنظمة العربية للترجمة
بناية «بيت النهضة»، شارع البصرة، ص. ب: 5996 ـ 113
الحمراء ـ بيروت 2090 1103 ـ لبنان
هاتف: 753031 ـ 753024 (9611) / فاكس: 753032 (9611)
e-mail: info@aot.org.lb - http://www.aot.org.lb

توزيع: **مركز دراسات الوحدة العربية**
بناية «بيت النهضة»، شارع البصرة، ص. ب: 6001 ـ 113
الحمراء ـ بيروت 2407 2034 ـ لبنان
تلفون: 750084 ـ 750085 ـ 750086 (9611)
برقياً: «مرعربي» ـ بيروت / فاكس: 750088 (9611)
e-mail: info@caus.org.lb - Web Site: http://www.caus.org.lb

الطبعة الأولى: بيروت، كانون الثاني (يناير) 2011

المحتويات

القسم الثاني: التخطيط

القسم الثالث: الهيكل التنظيمي

القسم الرابع: المراقبة والتحكم

القسم الخامس: دراسات الحالة

القسم السادس: ملاحظات ختامية



تقديم

سلسلة كتب التقنيات الاستراتيجية والمتقدمة
ضمن مبادرة الملك عبد الله للمحتوى العربي

يطيب لي أن أقدِّم لهذه السلسلة التي جرى انتقاؤها في مجالات تقنية ذات أولوية للقارئ العربي في عصر أصبحت فيه المعرفة محرِّكاً أساسياً للنمو الاقتصادي والتقني، ويأتي نشر هذه السلسلة بالتعاون بين مدينة الملك عبد العزيز للعلوم والتقنية والمنظمة العربية للترجمة ويقع في إطار تلبية عدد من السياسات والتوصيات التي تعنى باللغة العربية والعلوم ومنها:

أولاً: البيان الختامي لمؤتمر القمة العربي المنعقد في الرياض في 1428هـ ـ 2007 م الذي يؤكِّد ضرورة الاهتمام باللغة العربية، وأن تكون هي لغة البحث العلمي والمعاملات حيث نص على ما يأتي: (وجوب حضور اللغة العربية في جميع الميادين بما في ذلك وسائل التواصل، والإعلام، والإنترنت، وغيرها).

ثانياً: «السياسة الوطنية للعلوم والتقنية» في المملكة العربية

السعودية التي انبثق عنها اعتماد إحدى عشرة تقنية استراتيجية هي: المياه، والبترول والغاز، والبتروكيميائيات، والتقنيات المتناهية الصغر(النانو)، والتقنية الحيوية، وتقنية المعلومات، والإلكترونيات، والاتصالات، والضوئيات، والفضاء، والطيران، والطاقة، والمواد المتقدمة، والبيئة.

ثالثاً: مبادرة الملك عبد الله للمحتوى العربي التي تفعِّل أيضاً ما جاء في البند الأول عن حضور اللغة العربية في الإنترنت، حيث تهدف إلى إثراء المحتوى العربي عبر عدد من المشاريع التي تنفذها مدينة الملك عبد العزيز للعلوم والتقنية بالتعاون مع جهات مختلفة داخل المملكة وخارجها. ومن هذه المشاريع ما يتعلق برقمنة المحتوى العربي القائم على شكل ورقي وإتاحته على شبكة الإنترنت، ومنها ما يتعلق بترجمة الكتب المهمة، وبخاصة العلمية، ما يساعد على إثراء المحتوى العلمي بالترجمة من اللغات الأخرى إلى اللغة العربية بهدف تزويد القارئ العربي بعلم نافع مفيد.

تشتمل السلسلة على ثلاثة كتب في كُلٍّ من التقنيات التي حددتها «السياسة الوطنية للعلوم والتقنية». واختيرت الكتب بحيث يكون **الأول** مرجعاً عالمياً معروفاً في تلك التقنية، ويكون **الثاني** كتاباً جامعياً، والثالث كتاباً عاماً مُوجَّهاً إلى عامّة المهتمين، وقد يغطي ذلك كتاب واحد أو أكثر. وعليه، تشتمل سلسلة كتب التقنيات الاستراتيجية والمتقدمة على ما مجموعه ثلاثة وثلاثين كتاباً مترجماً، كما خُصص كتاب إضافي منفرد للمصطلحات العلمية والتقنية المعتمدة في هذه السلسلة كمعجم للمصطلح.

وقد جرى انتقاء الكتب وفق معايير منها أن يكون الكتاب من أمهات الكتب في تلك التقنية، ولمؤلفين مشهود لهم عالمياً، وأن يكون قد صدر بعد عام 2000، وأن لا يكون ضيِّق الاختصاص

بحيث يخاطب فئةً محدودةً، وأن تكون النسخة التي يترجم عنها مكتوبةً باللغة التي ألّف بها الكتاب وليست مترجمةً عن لغة أخرى، وأخيراً أن يكون موضوع الكتاب ونهجه عملياً تطبيقياً يصبّ في جهود نقل التقنية والابتكار ويساهم في عملية التنمية الاقتصادية من خلال زيادة المحتوى المعرفي العربي.

إن مدينة الملك عبد العزيز للعلوم والتقنية سعيدة بصدور هذه المجموعة من الكتب. وأود أن أشكر المنظمة العربية للترجمة على الجهود التي بذلتها لتحقيق الجودة العالية في الترجمة والمراجعة والتحرير والإخراج، وعلى حسن انتقائها للمترجمين المتخصصين، وعلى سرعة الإنجاز. كما أشكر اللجنة العلمية للمجموعة التي أنيط بها الإشراف على إنجازها في المنظمة وكذلك زملائي في مدينة الملك عبد العزيز للعلوم والتقنية الذين يتابعون تنفيذ مبادرة الملك عبد الله للمحتوى العربي.

الرياض 20 ربيع الأول 1431 هـ

رئيس مدينة الملك عبد العزيز للعلوم والتقنية

د. محمد بن إبراهيم السويل



مقدِّمة بقلم مانفريد بروي

لقد كان تطوير أنظمة البرمجيات وسيظل مُهمَّةً هندسية تواجهنا بالتحديات، وبالأخص بعدما أصبحت مشاريع البرمجيات متعددة الاختصاصات وموزَّعة. فقد صاغ فريتز باور (Fritz Bauer) مصطلح «هندسة البرمجيات» في مؤتمر عقد في غارمش (Garmisch) في أواخر الستينيات من القرن الماضي. ولقد شهدنا منذ أربعين سنة خلت تقدماً كبيراً في تحويل مهام هندسة البرمجيات إلى مهام أكثر وضوحاً ومدارة بشكل أفضل يمكن التنبؤ بنتائج اتباعها. وقد تحقق هذا بفضل التقدم الهائل في وضع نماذج لدورة حياة البرمجيات والتقدم في تحديد التقنيات الأفضل التي توظَّف في تخطيط وإدارة المشاريع والتصاميم المتطورة والمنهجيات الأكثر ملاءَمة وتقنيات النمذجة الممتازة والوسائل المحسِّنة. وقد أصبح لدينا اليوم إدراك أفضل ورؤية أعمق لأحكام ومبادئ وعوامل نجاح تطوير البرمجيات. ومع هذا، لا يزال هناك العديد من التحديات التي تواجه هندسة أنظمة البرمجيات الضخمة.

في الوقت الذي تتوزع فيه الشركات وشبكات الشركات وتتعاون حول العالم، يكون خيار القيام بالعمليات من خارج الحدود (off-shoring) أكثر شعبوية في مجال الإدارة، غير أن هذا الخيار يعتبر أمراً مثيراً للجدل في مجال هندسة البرمجيات. ويعتبر تطوير البرمجيات في المشاريع الموزَّعة دولياً، حيث يتم تطوير البرمجيات في عدة أماكن متباعدة عن بعضها بعضاً، أحد أهم تحديات العصر الراهن. ولا يزال يتعيّن علينا مواجهة تحديات تطوير البرمجيات الموزَّعة المراكز (GSD) والتغلّب عليها. وعلى الرغم من أن هناك العديد من المشاريع الصناعية الطموحة التي تستخدم منهجيات تطوير البرمجيات الموزَّعة المراكز، غير أننا لم نكتسب الخبرة الكافية عن المناهج التجريبية أو عن مبادئ أو أحكام تطوير البرمجيات الشاملة، على نحو منهجي، أو لم يتم تحليل النتائج بشكل ملائم أو حتى نشر هذه الخبرات.

تُعتبر هندسة البرمجيات مهمَّةً صعبة في كل الأحوال. ومع هذا، يتعيّن علينا في مجال تطوير البرمجيات الموزَّعة المراكز مواجهة المزيد من التحديات. وبعض هذه التحديات واضح ومعروف كاختلاف المناطق الزمنية مثلاً أو كصعوبات التواصل بين أعضاء الفريق الواحد لاختلاف الثقافات الاجتماعية التي ينتمون إليها. بينما يكون بعضهم الآخر أكثر دقة مثل بناء الثقة بين فرق العمل والاهتمام بعوامل النجاح العامة الرئيسة في مجال تطوير البرمجيات التي يصعب تحقيق الكثير منها في استخدام منهجيات تطوير البرمجيات الموزَّعة المراكز.

حتى الآن، تعتبر معرفتنا المنهجية عن تطوير البرمجيات الموزَّعة المراكز ضعيفة نسبياً. وهناك العديد من الأسئلة التي يتعيَّن الإجابة عنها في مجال تطوير البرمجيات الموزَّعة المراكز. وهذه الأسئلة تتعلق بالمواضيع الآتية:

- نماذج دورة حياة البرمجيات
- تخطيط المشاريع وتقويمها وإدارة المخاطر المتعلقة بها.
- الآثار المترتبة عن تطوير البرمجيات الموزَّعة المراكز على مرحلة تطوير البرمجيات، كهندسة المتطلبات، والتصميم، والتنفيذ، والتكامل، والاختبار.
- ضمان الجودة
- تنظيم فريق العمل
- مهارات فريق العمل والتدريب المتخصص
- توزيع العمل والمسؤوليات
- تنظيم عملية التواصل
- التأثير على المنتج البرمجي وبالأخص على الهيكلية
- البنية التحتية والوسائل الداعمة

بالإضافة إلى هذه المسائل التقنية والمنهجية المفصَّلة، هناك أسئلة أخرى تهمنا من وجهة نظر إدارية، مثل:

- ما هي عوامل وأسس النجاح؟
- ما هو الأثر الاقتصادي؟
- ما هي المشاريع التي يناسبها مبدأ تطوير البرمجيات الموزَّعة المراكز أكثر من غيرها؟

لذا قام قسم البحوث في شركة سيمنز في برينستون بإجراء التجارب في مجال تطوير البرمجيات الموزَّعة المراكز عام 2003 ضمن مشروع الأستديو العالمي (GSP)، وهذا مدعاة للسرور ويعتبر ذا قيمة في حد ذاته. ولقد أجريت تجربتان، منذ ذلك الوقت، مدة كل منهما عام واحد. وقد اشتركت العديد من الجامعات حول العالم في التجربة الأولى في العام المدرسي 2004/ 2005، وأما التجربة الثانية فقد أُجريت بعد عام واحد.

لقد كانت الأهداف من وراء هذه التجارب واضحة. فقد هدف المشروع إلى تحصيل البيانات التجريبية في تطوير البرمجيات الموزَّعة المراكز بالإضافة إلى العثور على العمليات والأدوات وطرق التعاون المناسبة. وقد ساهمت هذه التجارب بزيادة خبرتنا في كيفية تنظيم مشاريع تطوير البرمجيات الموزَّعة المراكز حتى نكفل نجاحها. وقد كانت طرق تخطيط التجارب واجراؤها أقلَّ وضوحاً في مرحلة التطوير الأولية عام 2003.

في الواقع، من المهم إجراء هذه التجارب من دون الأخذ بالاعتبار المخاطر الاقتصادية الهائلة التي تتعرض لها الشركات حين قيامها بالخطوات الأولى في تطوير البرمجيات الموزَّعة المراكز وتطبيق هذه المنهجية على برنامج خاص بالحياة اليومية. لذا، فإنه لا يتم تقويم تجربة مشروع من هذا النوع بطريقة ملائمة. ومن المهم في مشروع كهذا، جعل كافة البيانات معلنة ـ بعكس مشاريع تطوير البرمجيات الفعلية حيث نواجه مشكلة إبقاء حالات القصور سراً وعدم التحدث عن تفاصيل البرنامج السرية.

ما يمكنني قوله عن مشاركتي في مشروع الأستديو العالمي بطرق عديدة ورؤية مجموعة البحث الخاصة بي، وهي تشارك في التجربة لمدة عامين في جامعة ميونخ التقنية، هو أنها كانت تجربة ممتعة جداً وقيِّمة للغاية. وكان من المشجّع أيضاً رؤية الباحثين والطلاب المشتركين في المشاريع يكتسبون الكثير من الخبرة في تطوير البرمجيات الموزَّعة المراكز، ويصبح لديهم إدراك أوسع عن المسائل المتعلقة بتطوير البرمجيات.

نقدّم في هذا الكتاب مجموعة ممتازة لحصيلة عامين من النتائج والاستنتاجات لمشروع الأستديو العالمي. كما إنه يحوي الأساس الأولي للبيانات لتطوير البرمجيات الموزَّعة المراكز. ومع ذلك، فإن هذا الكتاب لا يقوم فقط بإعطاء تقارير عن مشروع الأستديو

العالمي، بل ما يفوق ذلك أهمية هو أنه يقدم عدة ملاحظات ونصائح عن كيفية تنظيم مشاريع تطوير البرمجيات الموزَّعة المراكز مع الأخذ بعين الاعتبار القيام بتطوير البرمجيات الموزَّعة المراكز بنجاح. وتعتمد هذه النصائح على حصيلة الخبرة في مشروع الأستديو العالمي، والتحليل العميق ومعرفة خلفية الباحثين المشتركين. وقد ثبت هنا مدى ضرورة توفر الخبرة لدى فريق العمل والباحثين والتدريب طويل الأمد الذي خضعوا له في مركز أبحاث شركة سيمنز في برينستون في مجالات العمل، والمتطلبات الهندسية، وفي التصميم والمنهجية والأدوات الداعمة.

ما أحبه في هذا الكتاب هو أنه عرض سهل لعدد من المواضيع المعقّدة والصعبة. ويعتبر هذا الكتاب ذا قيمة لمن له اهتمام بتطوير البرمجيات على هذا النحو، ولكنه يعتبر، بالأخص، ذا قيمة عظيمة للباحثين والمطورين ومديري المشاريع المهتمين بتطوير البرمجيات الموزَّعة المراكز والمشاركين فيها.

ويسرّني أن أشكر، على وجه الخصوص، فريق عمل مركز أبحاث شركة سيمنز على جهوده لإتاحة الفرصة لمجموعات البحث والعمل على نشر نتائج تجاربها المثيرة، واعتبار ذلك مساهمة منه في بناء المعرفة المتعلقة بهندسة البرمجيات. وإنه ليسرني، بصفتي أحد المشاركين في التجارب مع فريق عمل من جامعة ميونخ التقنية، أن أرى هذا الكتاب يخرج إلى النور. ويعتبر هذا الكتاب وثيقة قيِّمة ومساهمة مهمة في المواضيع التي تحظى باهتمام متزايد، والتي لها أثرها في الاقتصاد العالمي وفي المهارات الهندسية الخاصة بنا.

مانفريد بروي (Manfred Broy)
رئيس هندسة البرمجيات والأنظمة
كلية تكنولوجيا المعلومات
جامعة ميونخ التقنية



مقدِّمة بقلم جيمس هيربسليب

يبذل الفيزيائيون المتخصصون بعلم الجزيئات الكثير من الوقت والجهد والمال لصدم الجسيمات صدماً عنيفاً بعضها ببعض. ولا بد أن لديهم أسباباً منطقية لفعل ذلك ـ فعادةً لا يكون من الممكن مراقبة السلوكيات المهمة عندما تكون الجسيمات في بيئتها الطبيعية مترابطة مع بعضها بعضاً. إذ تساعد التصادمات القوية على بعثرة الجزيئات، وبعثرة الطاقة في كافة الاتجاهات. وبمراقبة الجزيئات في وضعية التشتت وبطاقة عالية جداً، يكون هناك فرصة أفضل للاستدلال على القوى الخفية التي كانت كامنة طوال الوقت.

تقوم المشاريع الموزَّعة جغرافياً ـ وبشكل غير مقصود ـ بوظيفة مشابهة في مشاريع تطوير البرمجيات. ونحن كممارسين وباحثين، لا نكفُّ عن تعلّم المهارات الضرورية والمتنوعة وبشكل سريع التي توفرها المشاركة الفعلية في الموقع وفرص التواصل والتعاون والمحتوى المشترك الذي تقدمه. لقد بدأنا ندرك جميعاً أن الفرق الموزَّعة جغرافياً تأخذ مسارات غير متوقعة، إذ لم نكن نعي أهمية «عمل

التنسيق» الذي يتطلبه المشروع، وكيف يتم هذا التنسيق بشكل غير ملحوظ في حال كان الأفراد يعملون، ببساطة، معاً في مكان واحد.

لا يُقصد مما سبق تزويد القارئ باستعارات مبالغ فيها أو ترفيهه، ولكن المقصود القول بأن هناك طريقتين على الأقل لقراءة هذا الكتاب الممتاز الجديد. إحداهما هي قراءته كما هدف الكتّاب، كدليل لإدارة الصعوبات المتعلقة بالمشاريع الموزَّعة، وذلك بالاعتماد على الخبرة المكتسبة، ودمج العديد من التقنيات السليمة والراسخة أو الابتكار في حال عدم وجود وسائل متوفرة. فهم حين يصفون طريقة ما ـ وليس المقصود هنا طريقة معينة، فإنهم يجتهدون لإيضاح طريقة التعامل مع المشاريع الموزَّعة جغرافياً مع إعطاء أولوية خاصة للمراحل الأولى الصعبة. ويجمع الكتاب عناصر من منهجيات لا يتوقع المرء أن تتعايش سلمياً، كالعملية المنطقية الموحدة (ROP) ومنهجية سكروم (Scrum). ويدعو الكتاب إلى التنقّل بين اتجاهات قد تبدو في بادئ الأمر متناقضة ـ مثلاً أن تكون أكثر سرعة أو أكثر تشدداً ـ ولكنك حين تتعمق في المادة، وتطَّلع على الأفكار المطبقة في دراسة الحالة الموثّقة جيداً، ستدرك أن الحكمة من الكتاب أعمق بكثير من «أن تكون سريعة» أو أن «تتشدد في تنظيم عملياتك». فالكتاب يُشرك القارئ في التفكير حول ما الذي يجب أن يكون سريعاً، وأي المواضيع التي تحتاج إلى نظام متشدد ولماذا. وفي حين يتعرض الكثير من القرَّاء لإغراء تطبيقه مثل وصفات كتب الطبخ، فليس الهدف منه أن يكون وصفة متّبعة، ولكنه بالأحرى اقتراح نهج نجح معهم، بالإضافة إلى وصفات مدروسة تتلاءم مع الواقع وتتلاءم مع الأفخاخ السيئة على طول الطريق. وسواء قمت باتباع هذا النهج أو لم تتبعه، فإنك ستستفيد بالتأكيد من درس الجغرافيا.

أما ما يقودني إلى الطريقة الثانية لقراءة هذا الكتاب فهو كونه مصدراً للفهم المتعمق للعوامل المؤثِّرة في تطوير البرمجيات بشكل

أعم. وكعضو في مجموعة الباحثين المشاركين في مشروع الأستديو العالمي، فقد بدأت أدرك أن العديد من الأفكار التي قمنا بتطويرها واختبارها في هذه «التجربة» يمكن أن تفيد فرق العمل المتشاركة في الموقع بقدر ما تفيد الفرق الموزَّعة جغرافياً. لقد أظهرت تجاربنا أن الشبكة الاجتماعية، على سبيل المثال، هي وسيلة قيِّمة وعملية لمراقبة المشاكل التي يمكن أن تعترض التعاون بين الأفراد في أي مشروع كبير نسبياً، إن وُجد التشارك في الموقع أو لم يوجد. تعتبر التقنيات المستخدمة لتحقيق الوضوح في مهام العمل فائقة الأهمية للفرق الموزَّعة، غير أنه من المرجَّح أن تكون أيضاً مفيدة للغاية في حالات أخرى. لقد جعلني هذا المشروع أدرك أن التعاون هو المشكلة الرئيسة، وأن التشارك في الموقع هو فقط أحد الحلول المتوفرة ـ وإن كان أقواها. وفي حال كان التشارك في الموقع غير ممكن، وضّح المؤلفون كيفية التعويض عن ذلك بحلول أخرى تساعد على إدارة المشروع بنجاح.

يقدِّم هذا الكتاب توليفة مدروسة جيداً من خبرات المؤلفين ومن الأدب المنشور ومن البحوث التي أجراها فريق عمل مشروع الأستديو العالمي. سيجد القارئ أن الكتاب هو دليل مفيد للغاية وعملي، ودليل تفصيلي للوضع الحالي الحديث. وعلى الرغم من وجود العديد من المشاكل التي لا تزال تحتاج إلى حلول في هذا المجال، غير أن قراءة هذا الكتاب جعلتني أدرك مدى التقدم الذي وصلنا إليه.

جيمس د. هيربسليب (James D. Herbsleb)
معهد بحوث البرمجيات
الكلية الدولية لعلوم الحاسب الآلي
جامعة كارنيغي ميلون



مقدِّمة

انطلقت فكرة هذا الكتاب بالتزام مركز البحوث في شركة سيمنز (SCR) بالعديد من مشاريع تطوير البرمجيات الموزَّعة المراكز. وشركة سيمنز هي مؤسَّسة متعددة الجنسيات يعمل لديها ما يزيد على 30,000 مهندس برمجيات، ولها فروع في معظم الدول.

لقد وجدنا خلال العمل على هذه المشاريع، مجتمعين، بأن شركات سيمنز حول العالم لديها معلومات مهمة عن مشاريع تطوير البرمجيات الموزَّعة المراكز يمكن أن تكون ذات فوائد جمّة في حال تمّ توليفها إلى مجموعة منظمة من المعرفة.

ومع هذا رغبنا بدايةً في التحقق من مجموعة المعرفة هذه واختبارها وتوسيع قاعدتنا مع مجتمع البحث. وبالتالي، قام مركز البحوث في شركة سيمنز بإطلاق برنامج بحث عن مشاريع تطوير البرمجيات الموزَّعة المراكز عام 2003 بتعاون وثيق مع جامعة كارنيغي ميلون (Carnegie Mellon University) وكلية هارفرد للأعمال (Harvard Business School) والمعهد الدولي لتكنولوجيا المعلومات

في بانغالور، وجامعة مونماوث (Monmouth University)، وجامعة ولاية بنسلفانيا (Penn State University)، وجامعة ميونخ التقنية (Technical University of Munich)، والجامعة البابوية الكاثوليكية في ريو غراند دي سول (Pontifical Catholic University of Rio Grande de Sul)، وجامعة ليمريك (University of Limerick). إن هذا الكتاب هو حصيلة الخبرة المتراكمة التي اكتُسبت من عدد من مشاريع تطوير البرمجيات الموزَّعة المراكز في شركة سيمنز، ومن التجارب التي أجرتها مجموعات البحوث التابعة لها والمؤلَّفة من ثماني جامعات في خمس دول عبر أربع قارات. إن للحكمة ثلاث دعائم رئيسة هي: (1) أفضل الممارسات لخبرة المشروع التي جعلت هذا الكتاب متماسكاً، (2) تحليل أدبي مكثّف قام بتوسيع قاعدة الخبرة، (3) البحث التجريبي الذي لم يتم من قبل في مجال تطوير البرمجيات الموزَّعة المراكز والذي أمدّنا برؤى غير اعتيادية.

إستناداً إلى هذه الخبرة المتراكمة، قمنا بتحديد عدد من العوامل الرئيسة لنجاح مشاريع تطوير البرمجيات الموزَّعة المراكز. ففي حين تكون هذه العوامل مفيدة لكافة المشاريع، تكون قضية العمل والضرورة أعظم شأناً عندما يتم توزيع المشاريع جغرافياً عبر عدة مناطق زمنية، مع فرق بعيدة لم تعمل معاً من قبل، ومضاعفة بسبب اختلافات اللغة والثقافة. ولكل من هذه العوامل قمنا بتنظيم أفضل الممارسات وحددنا أطراً عملية لها مع إعطاء توجيهات كبرى عن كيفية عمل أفضل الممارسات في مشروع ما.

أثناء عملنا على إنجاز هذا الكتاب، لاحظنا أيضاً رغبة لدى المهندسين العاملين في مشاريع تطوير البرمجيات الموزَّعة المراكز لتبنِّي ممارسات تطوير البرمجيات الذكية التي تفضل العمليات الخفيفة بالحد الأدنى من الوثائق التي يمكن أن تتكيف ديناميكياً مع متطلبات

المشروع. وقد يعتقد المرء أن التطوير الموزع يمكن أن يحتاج إلى الكثير من الوثائق بسبب متطلبات التواصل المرتفعة. وإن بدت متناقضة، فقد قادتنا هذه الملاحظات إلى التفكير في الجمع بين هذين الاتجاهين.

يقدم الكتاب منظوراً فريداً من نوعه عن السرعة الشاملة، المفهوم الذي يصف التقاء حركتين ـ موزَّعة عالمياً وتطوير البرمجيات السريعة. لقد أدركنا أن الكثير من الممارسات السريعة تعمل بشكل أفضل في بيئة عمل تشاركية، لذا حاولنا الحث على اعتماد الممارسات السريعة حيثما لزم الأمر لتحتفظ بمزاياها وقياس المنهجية لتحقيق سياق تطوير البرمجيات الموزَّعة المراكز. وقد حققنا هذا عن طريق تطوير وتحديد تهيئة البنية التحتية والعمليات والآليات وأفضل الممارسات التي تتيح أفضل التبادلات بين النظام والعمليات السريعة. وينصبُّ التركيز بالأخص على تطورات المنتج التي تتطلب هيكلية برمجيات جديدة أو معدّلة. كما ينصب تركيزنا على المراحل الأولية للمشروع نظراً إلى أهميته لمشاريع تطوير البرمجيات الموزَّعة المراكز.

هذا الكتاب مخصص لممارسي هندسة البرمجيات وطلاب إدارة المشاريع المهتمين بالتطوير الموزع للبرمجيات. كما سيكون مفيداً بالأخص لمديري المشاريع الذين يخططون لمشروع تطوير موزع، ولكن يمكن أن يجده المشاركون حالياً في مثل هذه المشاريع وسيلة مفيدة ومنظوراً جديداً عن هذا الموضوع. يمكن استخدام هذا الكتاب كملحق إضافي للدورات على المستوى الجامعي أو على مستوى التخرج في مجال إدارة مشاريع البرمجيات، وبالأخص حين مناقشة مواضيع محيطة بتطوير البرمجيات الموزع في عدة مناطق جغرافية.

لقد قسمنا هذا الكتاب إلى خمسة أقسام رئيسة. قسم المقدِّمة

المكوّن من الفصلين الأول والثاني، الذي يطرح دوافعنا من وراء هذا العمل ويصف إطار عملية لتطوير البرمجيات الموزَّعة المراكز. كما نعرض مجموعة من عوامل النجاح الرئيسة التي تكفل نجاح مشاريع تطوير البرمجيات الموزَّعة المراكز.

تشكل الفصول من الثالث إلى الثامن قسم التخطيط، حيث نبدأ بنماذج متطلبات الأعمال التي يتم صقلها تدريجياً بدءاً من ميزات المنتج إلى متطلبات النظام. وتزامناً مع متطلبات الجهد الهندسي، نقوم بتحديد متطلبات الهيكلية المهمة التي تؤثِّر في بنية النظام قيد النظر.

ومن ثم نأخذ بعين الاعتبار بنية وسمات الفرق الموزَّعة لقياس قدرات أفرادها على خلق نظام في ضوء الهيكلية المعطاة. وتُستخدم هذه المعلومات لاستنتاج وحدات النظام ومواصفاته. ويتم إعداد خطة البرمجة والإصدار طويلة الأمد من مواصفات متطلبات مزايا ووحدات النظام المطلوب بناؤه. ويتم في النهاية وضع الجدول الزمني لتنفيذ خطة البرمجة والتطوير وتحديد الموارد والمصادر والتكاليف، ويتم استخدام جدول زمني ذي منهجية عمل تكرارية لتحقيق التطوير المتزامن لعناصر النظام ضمن الفرق العديدة الموزَّعة في عدة مواقع حول العالم.

يكون لعدد من العوامل أهمية أكبر عندما يتم توزيع المشاريع في عدة مناطق جغرافية. فيصبح التعاون والإدارة أكثر صعوبة بسبب المسافة التي تفصل بين الفرق الموزَّعة في عدة مواقع. ففي حين يوجد هناك تقنيات ووسائل متاحة لتسهيل التواصل عن بُعد، فالبحوث التي قمنا بها تشير إلى أنها قد لا تكون فعالة بقدر الاجتماعات التي تتم وجهاً لوجه. لذا فإن إحدى طرق معالجة هذه المشكلة هي الحد من التواصل المفرط عن طريق تحسين درجة

التعاون بين الفرق. ويمكن تحسين التعاون عن طريق تقسيم النظام إلى أنظمة ومكوّنات فرعية حرة مقترنة. عندئذ يمكن لكل فريق العمل بشكل مستقل نسبياً عن الآخرين على نظام فرعي أو مكوّن معطى. ومع ذلك، ينبغي أن يكون هناك إدارة مركزية مسؤولة عن مواصفات هذه المكوّنات وبنيتها المشتركة وتنظيم تطورها الموزع عبر بناء منتظم للتطبيق القابل للاختبار. ويطلب كذلك من الأعضاء المباعدة بين الفرق المركزية والموزَّعة في عدة مواقع ليتمكنوا من (1) تخطّي المسائل المتعلقة بعدم معرفة الفرق بعضها بعضاً، أو لغات وثقافات بعضهم وغياب عمليات المشروع اليومية ليتابعها الأعضاء في المواقع المختلفة؛ (2) تكوين رؤية مشتركة للمشروع؛ (3) وبناء التعاون والثقة. ونناقش في الفصلين التاسع والعاشر بناء المؤسسة الذي لا يقوم فقط بتبني بناء الفرق، ولكنه أيضاً يحقق تعاوناً بنّاءً بين الفرق المركزية وتلك الموزَّعة في عدة مواقع.

يصف بند الرصد والمراقبة المسائل المتعلقة بجودة المنتج والعمليات في البيئة الموزَّعة جغرافياً. كما إننا ندرك أنه عندما تكون الإدارة المركزية هي المالكة للمنتج، وهي التي توزع تطوير مكوّناته الأساس على الفرق الموزَّعة في عدة مواقع، فينبغي أن تُؤخذ بعين الاعتبار العناية الشديدة والعلاقات طويلة الأمد مع هذه الفرق، ومعاملتهم كشركاء لها، وكمركز منافسة للمكوّنات التي تورّدها. يعتبر بناء هذه العلاقات مفيداً خصوصاً حين اعتبار احتياجات الإسناد والصيانة المستمرة للمنتج تحت التطوير. وبما إن التواصل يعتبر تحدياً كبيراً بين الإدارة المركزية والفرق الموزَّعة في عدة مواقع ـ وخاصة تلك التي تفصل بينها عدة مناطق زمنية ـ ينبغي أن يكون هناك اهتمام شديد بالبنية التحتية للوسيلة والكشف عن أنماط التواصل بين هذه الفرق لتحديد الاستراتيجيات الفعالة لإدارة هذا التواصل.

نقدِّم في الفصل الرابع عشر وحتى الفصل السابع عشر «دراسات حالة» تحصر خبراتنا (الجيدة منها والسيئة) من مشاريع متنوعة موزَّعة في عدة مناطق جغرافية قمنا بالاشتراك فيها مع شركة سيمنز. ولقد تم تشفير أسماء المشاريع في دراسات الحالة هذه لعزل هوية المنتجات والأشخاص المشاركين.

تتنوع المشاريع من حيث الحجم، مع الأخذ بعين الاعتبار أن المشاريع الصغيرة ستكون أكثر صلة بالشركات الصغيرة، بينما الكبيرة منها ستفيد المؤسسات الكبرى. إن الأجزاء ذات الصلة بدراسات الحالة هذه تكون مترابطة في كل فصول الكتاب، حتى تعطي للقارئ منظوراً عن كيفية تطبيق النظرية عملياً.

نقدِّم في الفصل الثامن عشر ملخصاً عن الكتاب، ونُلقي فيه الضوء على المسائل المتعلقة بمشاريع تطوير البرمجيات الموزَّعة المراكز. ونقوم في هذا الفصل بتحليل المشاريع الناجحة والفاشلة على حدٍ سواء، ونعيد سرد آرائنا المذكورة في أكثر من مكان في الكتاب عن كيفية تنفيذ مشروع ناجح بواسطة فرق موزَّعة في عدة مناطق جغرافية.

يقدِّم القرص المدمج المرفق عرض فيديو يعطي لمحةً عامةً عن مشروع الأستديو العالمي المذكور في الفصل الرابع عشر. إِضافةً إلى ذلك، تم تقديم (GSP Wiki) كمثال على بنية إدارة المعرفة المستخدمة في المشروع التجريبي لتطوير البرمجيات الموزَّعة المراكز.

شكر وعرفان

هذا الكتاب هو نتاج خبرات العديد من أصدقائنا وزملائنا الذين شاركوا في مشاريع تطوير برمجيات موزَّعة مراكزها في مناطق جغرافية متباعدة. ونحن مدينون لهم لمشاركتهم لنا بخبراتهم هذه.

ونود أن نسلّم بالعمل الممتاز الذي قام به الطلاب المشاركون في مشروع الأستديو العالمي، ونخص بالذكر زكريا الهدى (Zakaria El Houda)، وستيفان غيرسمان (Stefan Gersmann)، ولويز كوديرلي (Luiz Kuederli)، وألان مالون (Alan Malone)، وبرايان كيسي (Brian Casey)، وشارون إيد (Sharon Eade).

نقدم الشكر الجزيل للمدققين، المعروفين منهم لدينا وغير المعروفين لملاحظاتهم المهمة. ونحن ممتنون بشكل خاص لباتريك كايل (Patrick Keil)، وروبرت شفانك (Robert Schwanke)، وستيفن ماستيكولا (Stephen Masticola)، وجورج فونكس (George Phoenix)، وسكوت كارني (Scott Carney)، ووليام شيرمان (William Sherman).

وقد سرّنا العمل مع دار أورباخ للنشر في تطوير هذا النص.

ونخص بالشكر الخاص المحرر، السيد جون فيزاليك (John Wyzalek)، على توجيهاته وتشجيعه المستمر لنا طوال الوقت.

وأخيراً نشكر عائلاتنا لحبهم، وصبرهم، ودعمهم لنا أثناء العمل على إعداد هذا الكتاب. ونحن نهدي هذا الكتاب لهم بكل مَوَدَّة وحب.

القسم الأول

تمهيد



الفصل الأول

الــــدوافــــع

تسارعت في الآونة الأخيرة وتيرة الانتقال إلى تطوير البرمجيات الموزَّعة المراكز عالمياً. وتعكس عملية التوزيع الجغرافي لمشاريع تطوير البرمجيات في مناطق جغرافية مختلفة، قامت بها شركة سيمنز وشركات برمجة أخرى كبيرة وصغيرة، هذا الاتجاه. فقد بذلت شركة سيمنز الكثير من الجهود في عملية تطوير العديد من البرمجيات باستخدام فرق عمل موزَّعة، وتوظيف مفهوم أن هناك تأثيراً كبيراً لعملية توزيع وتطوير البرمجيات في دورة الحياة العملية، بما في ذلك الممارسات الإدارية (Herbsleb [et al.], 2005). وتتضمن عملية تطوير البرمجيات القابلة للتوزيع عالمياً (أو تطوير البرمجيات الموزَّعة المراكز) العديد من الإيجابيات والسلبيات التي قد تؤدِّي بأي مشروع تطوير برامج موزع إلى الخسارة ما لم يتم إدارته بحذر. ومن أهم الأمور المتعلقة بأي مشروع تطوير للبرمجيات الموزَّعة المراكز هو التردد بين السرعة والانضباط في العملية.

هذا الفصل هو مدخل عام عن تطوير البرمجيات الموزَّعة

المراكز ويعرض بعض التحديات المتعلقة بها، كما يعرض الحاجة إلى إدارة التعقيدات التي تعترض العملية بطرق فعالة وممنهجة بحيث تجنى الثمار المرجوة من تطوير البرمجيات الموزَّعة المراكز.

1-1 ما هو تطوير البرمجيات الموزَّعة المراكز

يمكن تعريف عملية تطوير البرمجيات الموزَّعة المراكز باختصار على أنها تطوير للبرمجيات توظف فرق عمل من مناطق جغرافية عديدة. وقد يكون، في بعض الأحيان، أعضاء فريق واحد من نفس الشركة أو المنظمة. وفي حالات أخرى قد يكون تعاوناً مشتركاً أو يُستعان بمصادر خارجية مختلفة. وقد تكون هذه الفرق في دولة واحدة (في الواقع، تشير الدلائل إلى أنه إذا تباعدت الفرق عن بعضها لأكثر من خمسين متراً، تكون المسافة أمراً غير مهم)، أو قد تكون في دول بعيدة عن بعضها بعضاً (Allen, 1984). إن وجود فرق العمل الخاصة بتطوير البرمجيات الموزَّعة المراكز ضمن مسافات فيزيائية متباعدة يسبب بعض التعقيدات والتحديات الممتعة التي أصبح العالم يتفهمها للتو.

لقد أدت العديد من العوامل التكنولوجية والتنظيمية والاقتصادية إلى نمو عولمة مشاريع التطوير. ومع أن ذلك كان يحصل منذ العقد الماضي أو قبل ذلك، فإن عملية العولمة أصبحت الآن أقرب إلى أن تكون القاعدة وليس الاستثناء. وفي ما يأتي بعض من أكثر المحفزات شيوعاً وأفضلها تأسيساً في تطوير البرمجيات الموزَّعة المراكز (Carmel, 1999; Herbsleb [et al.], 2005):

- القوى العاملة المدربة محدودة في مجال التكنولوجيا وهي إلزامية لبناء النظم المركبة المعقدة المتطلبة في الوقت الراهن.

- التفاوت في تكاليف التطوير، ما أدى إلى اللجوء إلى توزيع فرق العمل في عدة مناطق.
- أصبحت أنظمة العمل التي تعتمد على المناوبات أسهل وأيسر بسبب اعتماد الفروقات الزمنية بين المناطق ما أدى إلى تقصير المدة اللازمة لإيصال البرمجيات المنتجة إلى الأسواق.
- التقدم في مجال البنية التحتية (مثل عرض النطاق الترددي المتوفر للإنترنت وأدوات تطوير البرمجيات وتكاملها).
- الرغبة في الاقتراب ما أمكن من السوق المحلية.

في حين تختلف الصورة والحوافز الخاصة بمشاريع تطوير البرمجيات الموزَّعة المراكز، تبقى إحدى صفات المشاريع أمراً ثابتاً. فمن الصعوبة تنسيق المشاريع التي تكون فيها فرق العمل متباعدة في مناطق جغرافية مختلفة. بينما في مشاريع تطوير البرمجيات المنظمة التقليدية، عادة ما تكون هناك ممارسات قياسية وتكون مخرجاتها ومعالمها معروفةً، كما تتوفر لها قوالب مخصصة. غير أن بعض حالات التنسيق تتم بطريقة طارئة غير مجهَّز لها مسبقاً. ويبدو أن مهندسي البرمجيات قد اكتسبوا فهماً حدسياً لمتطلبات البرمجيات عن طريق التناضح. ويعيش تصميم النظام «الحقيقي» في عقول المصممين والمبرمجين وليس على الورق. وتشير الدلائل إلى أن الكثير من هذا «الفهم المشترك» يُشتق من العمل معاً بشكل متقارب. إن القدرة على تبادل الحديث في مقر العمل بالقرب من مبردات الماء أو أثناء استراحة الغداء مثلاً يساعد في تطوير مثل هذا الفهم المشترك بشكل كبير. إذ يتمكن زيد من فهم طريقة تفكير ليلى في ما يتعلق بالمشروع ويمكنه أن يترجم ملاحظاتها وأفكارها ويتبادل معها الآراء. فقد أدركت المنهجيات السريعة وحاولت دعم هذا النوع من التفاعل من خلال تطبيق بعض

الممارسات كالبرمجة المزدوجة(*) (pair programming) والاجتماعات اليومية السريعة(**) (standup meetings) والتنظيم بين العملاء والمبرمجين.

1-2 التحديات التي تعترض مشاريع تطوير البرمجيات الموزَّعة المراكز

تختلف مشاريع تطوير البرمجيات الموزَّعة المراكز في أشكالها وأحجامها. فليس مستغرباً اشتراك عدد من الأقسام والشركات والثقافات واللغات في مشروع واحد. وغالباً ما تجد بعض المشتركين في المشروع الواحد ممن لم يتقابلوا مع بعضهم البعض من قبل، وقد تتفاوت مستوياتهم وخبراتهم. وقد تكون لديهم دوافع تتعارض مع أهداف المشروع. تساهم هذه العوامل مشترِكة في جعل التنسيق بين الفرق وإدارة التطور ومراقبة التقدم في المشروع أكثر صعوبة (Herbsleb and Moitra, 2001; Mockus and Herbsleb, 2001). أظهرت الدراسات السابقة أن إتمام مهام العمل كان يستغرق وقتاً أطول بمرتين ونصف حين استخدام الإعدادات الموزَّعة بدلاً من استخدام الإعدادات المنتظمة (Herbsleb and Mockus, 2003).

هناك اعتقاد راسخ بأنه ما لم يتقابل الأشخاص المعنيون بالمشروع وجهاً لوجه، فقد يواجهون أوقاتاً عصيبة في تنسيق المهام

[الهوامش المشار إليها بـ(*)، هي من وضع المترجم]

(*) هو أن يعمل مبرمجان معاً على جهاز حاسوب واحد، بحيث يتساعدان ويتشاركان في التفكير والتحليل والتنفيذ والتدقيق، وهو ما يسرع من عملية البرمجة ويضمن الدقة في التنفيذ.

(**) تعقد هذه الاجتماعات بحضور جميع أفراد فريق العمل لمراجعة الإنجاز ومناقشة أي أمور أخرى تخص العمل؛ وتعقد لضمان عدم استهلاك الوقت في نقاشات جانبية أو توسيع نطاق النقاش، وهو ما يُعتبر خسارة في الوقت المخصص للمشروع.

المعَقَّدة عن بُعد، وخبرتنا تدعم هذا الاعتقاد. فحين ظهور تساؤل أو أمر معيّن يصعب تحديد الشخص الذي سيتم اللجوء إليه. فأي شخص يوجد في موقع بعيد لن يكون مطّلعاً على سير الأنشطة التي يقوم بها أعضاء الفريق الآخرون الذين يوجدون في موقع غير موقعه. لذا، يتم تبديد الكثير من الوقت في محاولة إيجاد الشخص الذي يستطيع المساعدة. الأمر الآخر الممكن حدوثه هو أنه في حال عدم توفر المعلومات، يتم اللجوء إلى الافتراضات. وقد تتعارض هذه الافتراضات مع متطلبات العمل أو مع الافتراضات الأخرى التي تقررت في موقع آخر، وهو ما يسبب المشاكل في المشروع.

عندما يتسلم مهندسو البرمجيات استعلامات أو تقارير تفيد بوجود مشاكل من أشخاص لا يعرفونهم شخصياً، فإنهم سيكونون أقل حماسة للاستجابة. فقد يترجمون التقرير حول المشكلة على أنه نقد لعملهم، أو قد لا يشعرون بأهمية المشكلة التي تواجه زميلهم الذي يوجد في موقع بعيد. فقد لا يستجيبون أبداً للرسائل الإلكترونية أو إذا ردوا عليها قد لا تكون ردودهم بنفس البراعة التي قد تتصف بها ردودهم في حال كانوا يعرفون الأشخاص الذين أرسلوها. علاوة على ذلك، فقد لا يتوفرون للمساعدة نظراً إلى وجود عطل رسمية محلية لا يُحتفل بها في دولة الشخص الذي أرسل الطلب أو الاستعلام أو التقرير. كما إنه ليس من المستغرب أن تختلف الدوافع بشدة من فريق إلى آخر. وقد يخشى الفريق الموجود في أحد المواقع من أن المشروع ـ في حال نجاحه ـ سيحل محل منتج قديم كانوا مسؤولين عنه مسبقاً. وقد يمنع خوف فقدان الوظيفة الأشخاص في ذلك الموقع من مشاركة المعلومات التي يملكونها مع غيرهم بأريحية.

هذا، وتلعب عوامل فوارق الثقافة واللغة دوراً في هذا المجال.

فعلى سبيل المثال، خبرنا وجود مقاومة ونفور من بعض الأشخاص غير المتحدثين باللغة الإنجليزية للمشاركة بفعالية في الاجتماعات التي تتم عن بُعد. فهم يفضلون توجيه أسئلتهم وما يعترضهم في المشروع عن طريق الرسائل الإلكترونية. وقد وجدنا (في وقت من الأوقات كنا مجموعة من الأميركيين) أن الرسائل الإلكترونية بطيئة ومحبطة. وقد تم اكتشاف ذلك ووُضعت بروتوكولات واضحة للتعامل مع الإشكاليات وذلك بعد فترة مهمة في هذا المشروع. كما قد تسبب الثقافة ونمط الحياة المحليان بعض الإرباك والخيبة. فعلى سبيل المثال، تكون معرفتنا جيدة عندما يتعلق الأمر بثقافة تمتُّ لها بصلة مباشرة، بينما يختلف الأمر حين التعامل مع ثقافة أخرى تختلف عن ثقافتنا (نتجنب مراقبة التفاصيل). وأما في المراسلات الهاتفية والبريد الإلكتروني، فقد ظهر أن الثقافة المباشرة تسبِّبُ إحباطاً بحيث يبدو النظراء غير قادرين على الفهم. فقد يتصلون هاتفياً ويسألون عن مواضيع ليست ذات علاقة قبل أن يفهموا المطلوب. كلما قل الأثر المباشر للثقافة، فقد يُنظر إلى النظراء على أنهم جاهلون. وتم حل هذه المفاهيم الخاطئة بعدما تم تنظيم سلسلة من الاتصلات الشخصية، وأصبح التفاني والجدية التي كانوا يعالجون بها المشاكل واضحاً، وتشكلت العلاقات الشخصية بينهم حين التقائهم مساءً لشرب القهوة.

تنتهي العديد من القضايا اللوجستية الأخرى باستغراق وقت أكثر من المتوقع. ومن الصعوبات والقضايا التي تستغرق وقتاً وتستنفد الموارد القيِّمة ووقت المشروع محاولة معرفة كيفية تنظيم البنى التحتية التقنية والتعامل مع قضايا الربط التي تفرضها طبوغرافيا الشبكة الخاصة ومعرفة كيفية تنظيم الممارسات الإدارية والعمليات التنظيمية حيثما لزم الأمر، إلى غير ذلك. وإذا لم تؤخذ العوامل المذكورة

أعلاه في الحسبان ولم يتم معالجتها، فإن احتمال إنتاج منتج ذي جودة ضمن الموازنة والجدول الزمني المحددين يُقلَّص بشكل كبير (في الواقع، إن الانتباه إلى هذه العوامل لا يضمن نجاحها).

1-3 إدارة تطوير البرمجيات الموزَّعة المراكز

لقد قمنا بتطوير مجموعة من الممارسات التي نستخدمها في المشاريع التي تتوزع جغرافياً، وذلك بعد معاناة طويلة واعتماد على كثير من حالات التجربة والخطأ. وهذه الممارسات ليست الحل السحري، لكننا حاولنا جمع الخبرات الجيدة والسيئة على حدٍّ سواء وتصنيفها في إطار عمل يوفِّر طريقة أكثر تنظيماً لإدارة وتنفيذ مشاريع تطوير البرمجيات الموزَّعة المراكز.

قمنا من خلال هذه الخبرة التجميعية بتحديد العوامل المهمة لنجاح مشاريع تطوير البرمجيات الموزَّعة المراكز، وصنفنا أفضل الممارسات لتوظيف هذه العوامل للحصول على نتائج ناجحة. وقمنا بتطوير إطار عملية تقدم أفضل الممارسات المصنفة في عوامل النجاح الحاسمة. ويتركز هذا الإطارعلى نموذج البرمجة السريعة العالمية الذي يجمع بين الاتجاه نحو عملية تطوير البرمجيات الموزَّعة المراكز وممارسات البرمجيات السريعة. وحين التفكير في ممارسات البرمجة السريعة، يُتوقع من القارئ أن يستحضر في ذهنه صورة لتطوير البرمجيات المستخدمة أساساً في بيئات العمل المنتظمة. لقد أدركنا أن العديد من ممارسات الطريقة السريعة (Cockburn, 2002)، قد تخطت ذلك نحو تحديد المشاكل ذات المفهوم المشترك وتطور المشروع. هذا وتكون هذه الممارسات قصيرة حين استخدامها في سياق البرمجة العالمية. وقد حاولنا زيادة الممارسات السريعة حيثما لزم الأمر بحيث يمكننا الحفاظ على معظم

المنافع التي تتحقق منها مع قياس طريقة البرمجة في سياق تطوير البرمجيات الموزَّعة المراكز. ويمكن حدوث ذلك فقط عن طريق تطوير وتحديد الإعدادات الأساسية والعمليات وآليات العمل وأفضل الممارسات التي تسمح بالحصول على أقصى تبادل بين نظام السرعة ونظام العملية (Boehm and Turner, 2004).

يرتكز إطار عمل العمليات خاصتنا على النموذج التكراري (iterative model) للبرمجة مع تعريف واضح للأدوار والمسؤوليات وتعريف البنية التحتية لتسهيل التنسيق والتعاون والجوانب الأخرى التي تربط الأفكار العامة لنظام عملية تطوير البرمجيات الموزَّعة المراكز بالسرعة. إن فهم معتقدات العمل بكفاءة في بيئة موزَّعة هو ما يُشكِّل أساس منهجيتنا ـ كتوزيع العمل على عدة مواقع بأمثل طريقة وزيادة وتحسين طرق التواصل بين هذه المواقع وتسهيل الوصول إلى الخبراء باستخدام الأدوات والممارسات لتحسين الوعي والإدراك خلال هذه المواقع.

1-4 الملخص والاستنتاجات

إن عملية تطوير البرمجيات الموزَّعة المراكز هي عملية صعبة بطبيعتها؛ إذ تعترضها مشاكل في التنسيق والدوافع واللحاق بركب التقنيات المحدثة والبنى التحتية والعمليات التي قد تسبب توقف المشاريع. لكن تطوير البرمجيات الموزَّعة المراكز حقيقة واقعة وظهرت لتدوم، لذا علينا أن نتعلم كيفية تنفيذ مثل هذه المشاريع بكفاءة ونجاح.

صُنّف هذا الكتاب كحصيلة سنوات عديدة من الخبرة في مجال مشاريع تطوير البرمجيات الموزَّعة المراكز إلى مجموعة من الممارسات التي يمكن تطبيقها في مشاريع كبيرة وصغيرة تستخدم فرق عمل موزَّعة في مناطق جغرافيَّة عديدة.

5-1 أسئلة للمناقشة

1. لماذا تُعتبر إدارة مشاريع تطوير البرمجيات الموزَّعة المراكز أصعب من إدارة المشاريع المنتظمة؟
2. أي ممارسات برمجة سريعة أسهل تطبيقاً، وأيها يشكل تحدياً في مشاريع تطوير البرمجيات الموزَّعة المراكز؟

المراجع

Books

Allen, Thomas John [et al.]. *Managing the Flow of Technology: Technology Transfer and the Dissemination of Technological Information within the R&D Organization*. Cambridge, MA: MIT Press, 1984.

Boehm, Barry and Richard Turner. *Balancing Agility and Discipline: A Guide for the Perplexed*. Boston, MA: Addison-Wesley, 2004.

Carmel, Erran. *Global Software Teams: Collaborating Across Borders and Time Zones*. Upper Saddle River, N. J.: Prentice Hall, 1999.

Cockburn, Alistair. *Agile Software Development*. Boston, MA: Addison-Wesley, 2002.

Periodicals

Herbsleb, James D. and Audris Mockus. «An Empirical Study of Speed and Communication in Globally Distributed Software Development.» *IEEE Transactions on Software Engineering*: vol. 29, Issue 6, June 2003, pp. 481-494.

Herbsleb, James D. and Deependra Moitra. «Global Software Development.» *IEEE Software*: vol. 18, no. 2, 16-20 March/April 2001.

Conferences

Herbsleb, James D., Daniel J. Paulish and Matthew Bass. «Global Software Development at Siemens: Experience from Nine Projects.» *Proceedings of the 27th International Conference on Software Engineering*, St. Louis, MO, 2005, pp. 524-533.

Mockus, A. and James D. Herbsleb. «Challenges of Global Software Development.» *Proceedings of the Seventh International Software Metrics Symposium,* London, England, 4-6 April 2001, pp. 182-184.

الفصـــــل الثاني

عوامل النجاح الحاسمة في تطوير البرمجيات الموزَّعة المراكز

يعمل جون لدى شركة باس (BAS corporation) التي نمت وتطورت على مر السنين عن طريق اندماجها مع شركات أخرى حول العالم. وترغب شركة باس في تعزيز المنتجات المتفاوتة من الشركات المفردة في خط إنتاج واحد، وذلك كي يبسّط تطوير المنتج الجديد وتحقيق فوائد اقتصادية. وقد طُلب من جون أن يبذل ما في وسعه لتحقيق ذلك. ومع أنه استطاع إنجاح العديد من المشاريع التي شكَّلت تحدياً بالنسبة إليه، غير أنه لم يعمل قط في مشروع يتضمن تنسيق جهود فرق البرمجة من عدة مواقع حول العالم. يبدأ جون بالتساؤل عن كيفية اختلاف طريقته في العمل في هذا المشروع عن إدارة مشروع منتظم في موقع واحد.

في عصر الاقتصاد العالمي، يجد العديد منا أنفسهم في أوضاع وظروف متشابهة لأسباب مختلفة. فأجور العمل منخفضة حالياً في

دول أوروبا الشرقية والبرازيل والهند والصين، وقد يكون هناك توفير في التكلفة إذا تم اللجوء إلى مصادر خارجية لتطوير البرمجيات في هذه الدول. فقد تتوفر لدى الشركة في مجموعة من التقنيات أو في مجال معين، فتحاول استغلال ذلك لجذب الشراكات. وتحت وطأة نقص الموظفين والضغط الذي يتسبب عن ضرورة الالتزام بمواعيد التسليم، تُجبر الشركة على تنفيذ عدد من مشاريع تطوير البرمجيات في ذات الوقت باستخدام قوى عاملة من دول مختلفة.

نتناول في هذا الفصل بعض الأمور المتعلقة بمشاريع تطوير البرمجيات الموزَّعة المراكز ويُبرز العوامل الحرجة والحاسمة لنجاح هذه المشاريع. إضافة إلى ما تقدم، يبدأ الفصل بشرح إطار عمل عملية إدراك عوامل النجاح هذه. وتشكل المقدّمة العامة في هذا الفصل خريطة الطريق للتفاصيل التي تتابع في الفصول الآتية.

2-1 قضايا

عندما تتعامل الشركات بمشاريع تطوير البرمجيات الموزَّعة المراكز، لا تقوم بتقويم أثر استخدام فرق العمل الموزَّعة في مناطق جغرافية عديدة. وأحد أسباب ذلك أن الحد الذي يعتمد فيه الأشخاص الذين يعملون في بيئة عمل منتظمة على التواصل غير المخطط له وغير الرسمي للبرمجة غير معرَّف. إن أثر عدم وجود هذا النوع من التواصل وعدم الأخذ بالحسبان غيابه أمر خطير بلا شك. فقد اختبر مؤلفو هذا الكتاب هذا الأثر السلبي في العديد من المشاريع الفاشلة في شركات عديدة متنوعة.

بناءً على هذه الخبرات، تم تعريف عدد من العوامل الحاسمة لنجاح مشاريع تطوير البرمجيات الموزَّعة المراكز. وهذه العوامل ذات مستوى متقدم باعتراف الجميع، وهي لم تكن وليدة الصدفة، بل هي نقل ملائم للخبرات المتولدة على مدى السنوات. وللمساعدة في

تفسير كيفية توظيف عوامل النجاح الحاسمة عملياً، قمنا بتطوير إطار عمل لتوضيح العملية التي تجسِّد هذه العوامل. ولا يُقصد بذلك اقتراح إطار العمل هذا كحل للجميع. إن مبدأ «حجم واحد يلائم الجميع» لا ينطبق هنا. لكن المقصود هنا هو توفير شرح أعمق وإظهار كيفية إدراك هذه العوامل في مشروع تطوير البرمجيات الموزَّعة المراكز.

2-2 عوامل النجاح الحاسمة

نناقش في هذا القسم عوامل نجاح مشاريع تطوير البرمجيات الموزَّعة المراكز. ففي حين أن هذه العوامل مفيدة لجميع المشاريع، غير أن حالة العمل والضرورة أكثر أهمية عندما تكون المشاريع موزَّعة في عدة مناطق جغرافية تختلف فيها المناطق الزمنية وتعمل فيها فرق العمل وتتعامل مع بعضها بعضاً عن بُعد، من دون أن يكون لدى أي منها تاريخ عمل مع الآخرين، إضافة إلى فروقات اللغة والثقافة. تهدف هذه العوامل إلى تجسيد الخبرة المكتسبة من العديد من هذه المشاريع (وهي بالتالي تكون نظرية جداً). يعتمد إدراك هذه العوامل في أي مشروع على أمور كثيرة كالعمليات التنظيمية وخصائص فرق العمل والبنى التنظيمية والنظام الذي يجب بناؤه. يجب أن ينظر إلى تطبيق عوامل النجاح هذه كاستثمار في التخفيف من حدة المخاطرة وضمان الجودة. في الفصول الباقية من هذا الكتاب، سنعطي أمثلة وتوجيهات على كيفية حساب هذه العوامل في مشاريعكم.

2-2-1 الحد من الغموض

في حين أن الحد من الغموض والالتباس المرتبط بجميع مظاهر أي مشروع أمر مرغوب فيه، يتم الكشف عن العديد من حالات عدم

التأكد والغموض في المشاريع المنتظمة، والتي تنتج من التواصل غير الرسمي. لا تتوفر هذه الرفاهية في مشاريع تطوير البرمجيات الموزَّعة المراكز. فالغموض يقود إلى الافتراضات. ولا تكون هذه الافتراضات ظاهرة للعيان بسهولة، وهو ما يؤدي إلى وجود بعض الافتراضات المتناقضة لبعض الوقت قبل أن تعبِّر عن نفسها كمشاكل. إن ظهور مثل هذه المشاكل يتطلب إعادة التخطيط وإعادة التصميم وإعادة العمل؛ وتنفيذ هذه الأمور ليس بالأمر السهل في بيئة عمل موزَّعة. فقد يكون التنسيق بطيئاً وشاقاً وغير فعال. كما إن تأثير وجوب وجود أنشطة التنسيق الكثيفة قد يسبب تأخيراً طويلاً كما يترك فرق العمل عاطلين ومحبطين، ويتسبب ذلك بمشاكل في الجودة وقد يتسبب في توقف المشروع (أكثر من مرة).

قد يرتبط الغموض والالتباس بالعمليات التنظيمية، أو بالممارسات الإدارية أو متطلبات العمل أو تصميمه. فإذا اجتمعت هذه الحالات مع نقص في الخبرة والمعرفة التي يمتلكها أعضاء فريق العمل، يصبح الوضع أكثر تعقيداً. لذا يتطلب الأمر الكثير من التفكير والعمل للاتفاق حول كيفية عمل الفرق المختلفة مع بعضها بعضاً لضمان استخلاص الهدف المطلوب من المشروع. يجب أن تعرِّف العمليات قواعد الارتباط بوضوح، كما يجب صياغة متطلبات العمل بحيث يسهل فهمها؛ ويجب تطوير الهيكلية وتوسيعها مع الملاحق بين النماذج المعرّفة والعناصر المحددة مع بينيات معرّفة جيداً. هذا ويجب أن يتم إنشاء رزم العمل للفرق الفردية متضمنةً مهام العمل؛ فضلاً عن كل ما تقدم، فحين إجراء التواصل بين الفرق للحصول على بعض التوضيح، يجب أن يضمن الشخص ذو العلاقة أن جميع إجابات الأسئلة صحيحة ومفهومة.

إن ما هو واضح لإحدى فرق العمل ذات خلفية وثقافة خاصتين

قد لا يكون بذات الوضوح لفريق آخر. وقد يتطلب الأمر وجود آليات للتحقق من فهم مظاهر محددة في المشروع ووجود برامج تدريب أو تبديل للأفراد. يجب أن يتم التوظيف بعناية وحذر للموازنة بين الخبرات التقنية والمعرفة في مجال العمل والمطلوب. تحكم خصائص المشروع مقدار الغموض الذي يمكن التعامل معه إضافةً إلى الآليات الملائمة لتحديد هذا الغموض. فعلى سبيل المثال، إذا ارتبطت إحدى المنظمات الراعية بعلاقة طويلة الأمد مع فرق العمل عن بُعد، ستتأسس على الأرجح ثقافة عمل تسهّل أمور التواصل الخاصة بالمشروع إذا ما قورن الأمر بوجود فريقَيْ عمل لم يتقابلا أو يعملا مع بعضهما البعض من قبل.

2-2-2 الحد الأقصى من الاستقرار

يؤثر عدم الاستقرار في عدد من جوانب المشروع. فعلى سبيل المثال، العمليات السريعة (agile processes) هي استجابة للمتطلبات غير الواضحة وغير المستقرة. إن من أهم المعتقدات المرتبطة بالعمليات السريعة أنها تسعى جاهدة إلى إنشاء بيئة تستخدم التواصل الطارئ وغير الرسمي بأفضل طريقة ممكنة كالبرمجة المزدوجة والعمل في نفس المكان ووجود من يمثِّل العميل في موقع العمل، والفترات البرمجية المكررة القصيرة (short iterations) والاجتماعات اليومية السريعة التي تتم وقوفاً، إلى غير ذلك. فمن المنطقي أن يكون الاستقرار عاملاً حاسماً في المشاريع الموزَّعة، التي تصعب فيها مثل هذه الأنواع من التواصل. إن تأثير وجود عوامل عدم استقرار في المشروع هو نفس تأثير وجود أمور غامضة غير واضحة في المشروع .

ماذا يعني ذلك؟ حسناً، ذلك يعني أنه يعتمد بدرجة كبيرة على

تفاصيل المشروع، لكن قد يعني ذلك وجوب تأخير البدء بمرحلة التطوير، فلا يتم البدء بها كما هو الحال في المشاريع المنتظمة التقليدية، وذلك لإتاحة الفرصة لاستقرار مواصفات متطلبات العمل والتصميم إلى أفضل وضع ممكن.

تُعتبر طلبات التغيير (change requests) مخاطرة كبيرة في مشاريع تطوير البرمجيات الموزَّعة المراكز عالمياً، إذ يتطلب تنفيذها وقتاً أطول بـ 2,4 مرة من الوقت الذي تتطلبه المشاريع المنتظمة (Herbsleb and Mockus, 2003). وبناءً على ذلك، لا يمكن التهاون في هندسة المتطلبات وهيكلتها بامتياز.

وقد يعني ذلك تطوير المزيد من النماذج البدائية (prototypes) وتصميم العديد من نماذج واجهات الاستخدام وتطوير أطر عمل مختلفة أو تخصيص بيئات برمجية. هناك العديد من الطرق التي يمكن اتباعها لتحقيق الاستقرار في مختلف جوانب المشروع. وسنعطي أمثلة ونشرح كيفية تحقيق ذلك لاحقاً في هذا الكتاب.

2-2-3 فهم الروابط

ينطوي الترابط بين المهام على حجم ونوع وتواتر التنسيق المطلوب بين المشاركين في مشروع تطوير البرمجيات الموزع. فهي مهمة وحاسمة لوضع خطة المشروع وتنفيذه للتعامل مع احتياجات عملية التنسيق. وعادةً ما تُعامَل الروابط على أنها «تستدعي العلاقة calls relationship» بين النظم الفرعية التي يتعين توزيعها. ففي حين أن مثل هذه العلاقة لا تعني وجود ترابط، غير أن هناك جوانب أخرى يجب أخذها بعين الاعتبار. فمن وجهة النظر التقنية، هناك العديد من العوامل التي قد تستلزم التنسيق. على سبيل المثال، إذا كان هناك متطلَّب صعب مستتر فقد يتطلب ذلك تنسيقاً بين جميع

فرق العمل ذات العلاقة بالبرمجة (coding) التي يمكن تنفيذها في نفس الوقت؛ أو إذا كانت إحدى الفرق مسؤولة عن تطوير نظام فرعي يتطلب خوارزمية معالجة الصور، فقد يتطلب ذلك حاجة ملحة للتنسيق.

إن الجوانب التقنية للمشروع ليست المصدر الوحيد للروابط. إذ إن هناك العديد من الروابط المؤقتة التي تنشأ عن التخطيط أو عن تقسيم عملية التنفيذ إلى مراحل. فعلى سبيل المثال، قد يتم توزيع عملية التنفيذ في مراحل البرمجة أو بواسطة النماذج المكتملة أو بواسطة النظم الفرعية التي توجد في مواقع البرمجة المختلفة. وهذا الأخير قد يكون مفيداً أكثر من السابق. هذا، وسنقوم بتفصيل كيفية تحديد الروابط وأخذها في الحسبان في مشروعك.

4-2-2 تسهيل عملية التنسيق

إن استكمال فهم طبيعة المهام المترابطة أمر مهم لتيسير التنسيق المنسجم. يجب أن تكون فرق العمل التي تقوم بعملية التنسيق قادرة على عمل ذلك بطريقة تتناسب مع الاحتياجات. هناك طرق عديدة مختلفة يمكن لفرق العمل استخدامها للتنسيق. ففي حين يعتقد أن التنسيق ما هو إلا عملية اتصال، غير أنه في الواقع أوسع من ذلك بكثير. فالتواصل هو أحد السبل التي تُستخدم للتنسيق؛ وهناك طرق أخرى كتنسيق العمليات والممارسات الإدارية وهيكلية خط الإنتاج على سبيل المثال لا الحصر. ويمكن أن يُنظر إلى هذه الخيارات باعتبارها مفاضلة بين النفقات العامة والمخاطر. مثلاً، إذا طُلب من شركة برمجيات تصميم وتطوير إطار عمل يحد من خيارات التصميم لفرق عمل تعمل عن بُعد، بينما يُفرض على فرق العمل التي تعمل عن بُعد التكامل والاندماج مع الهدف الأصلي للفريق، تكون تكلفة

بناء إطار العمل كهذا مرتفعة جداً. كما سيحد إطار العمل من الحاجة لتوضيح قرارات التصميم.

من خلال تجربتنا، تبين أن مشاريع تطوير البرمجيات الموزَّعة المراكز عالمياً تركِّز في الكثير من الحالات على الحد من التكاليف ما أمكن، وذلك بنقل عملية البرمجة إلى مناطق أقلَّ تكلفة. بينما يُعطى القليل من الاهتمام إلى الاستثمار في تطوير العمليات وبيئة البرمجة التي يمكن أتمتتها لدرجة ما، وبالتالي توجّه جهود التنسيق بين فرق العمل المتعاونة إلى حد بعيد. لا تكون الحاجة واضحة دائماً منذ البداية، وهذا ينطبق على مقدار الجودة التي سيعمل بها برنامج ما في بيئة عمل معينة. إذاً من الحكمة اتباع استراتيجيات النسخ الاحتياطية (backup) التي يلجأ إليها إذا ما سارت الأمور على غير ما يرام.

2-2-5 الموازنة بين المرونة والتشدد

يجب أن تكون ممارسات مشاريع تطوير البرمجيات الموزَّعة المراكز أكثر مرونة وأكثر تشدداً من مثيلاتها من المشاريع المنتظمة. فغالباً ما يكون أعضاء فرق العمل التي تعمل عن بُعد ذوي خلفيات مختلفة ويتعاملون مع عمليات برمجة مختلفة، وهم يختلفون أيضاً في مستويات الخبرة والمعرفة في مجال العمل والثقافة، وتتَّبع في بعض الحالات ممارسات إدارية مختلفة. يجب أن تكون عملية البرمجة مرنة بشكلٍ كافٍ لتلائم هذه الفروقات. وقد يعني ذلك إعطاء فرق العمل التي تعمل عن بُعد بعض الحرية في اتِّباع منهجية داخلية للبرمجة أكثر سرعة.

يجب أن تكون العملية مبنية بطريقة أكثر تنظيماً وثابتة بطريقة معينة. ويجب أن تكون صلبة بما فيه الكفاية لضمان دقة تحديد

مظاهر المشروع وأن العمليات المتبعة كشبكات الأمن الطبيعية في مكانها، كما يجب ضمان فهم الأمور الأخرى كالتعليمات، وأن متطلبات العمل قد تحققت، وأن عمليات إدارة الإعدادات والتهيئة معرَّفة بشكل كافٍ، وأن إجراءات التكامل والاختبار مناسبة، إلى غير ذلك من أمور. هذه الأمور ضرورية لمراقبة تطور المشروع وضمان الالتزام بمواعيد التسليم وضمان الجودة.

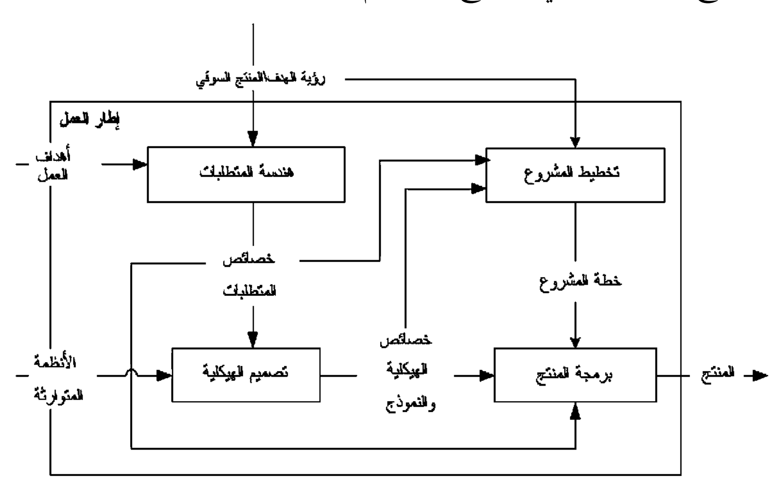
2-3 إطار العملية

يوضح الشكل 2-1 إطار عمل مشروع تطوير البرمجيات الموزَّعة المراكز. ليس الغرض هنا وصف العملية الدقيقة التي يجب أن يتبعها كل مشروع، لكن الهدف هو تعريف المفاهيم التي عرض لها في هذا الكتاب بطريقة أسهل وعرض كيفية إدراك عوامل النجاح الحاسمة؛ ولتحقيق ذلك يتم وصف إطار عمل متقدم مرفقاً بخطوات العمل التي يجب على فرق العمل التابعة للمشروع اتباعها. ولا نقصد وجود عملية مناسبة لجميع المشاريع. ففي حين أننا استخدمنا هذه العملية عملياً، غير أنها تتطلب تكييفاً واعتماد السياق المعطى. وتم توضيح الأنشطة التي يتضمنها إطار العمل كخطوات متتالية تعطي جوانب المتطلبات والهيكلية ومخطط المشروع وتطوير المنتج؛ ولكن عملياً، ستكون هذه الخطوات متكررة كثيراً.

إن الهدف من خطوة هندسة المتطلبات هو كسب فهم جيد للمتطلبات بحيث يعرف المرء المطلوب من تلك المتطلبات وتحديد المتطلبات القيادية (هي المتطلبات المهمة التي لها قيمة في العمل «business value» أكثر من غيرها في النظام الذي يتم تطويره).

نستخدم هندسة المتطلبات التي يحكمها نموذج ومنهجية البرمجة المتبعة (Berenbach, 2003; 2004a, and 2004b). التي تستخدم لغة

النمذجة الموحدة (UML) لتحديد المتطلبات. لا توفِّر لغة النمذجة الموحدة أي دليل على كيفية وجوب استخدامها؛ علينا ـ من خلال عمليات البحث ـ تطوير منهجيات تؤدي قدرة لغة النمذجة الموحدة على إضافة معنى للروابط التي تربط بين المتطلبات، وبالتالي تسهيل عملية تحليل المتطلبات من حيث الإتمام والاتساق والجودة، وكذلك السماح بالتعقب الآلي لنماذج التصميم والاختبار.



الشكل 2–1: إطار عمل لمشروع تطوير البرمجيات الموزَّعة المراكز

تُعتبر هندسة متطلبات العمل التي يحكمها نموذج البرمجة عملية تطوير وصيانة منهجية لمتطلبات المنتج التفصيلية والتي تبدأ عندما يكتسب المنتج دفعاً كافياً؛ وهناك حاجة مبكرة في المشروع لتطوير حالة عمل وتعريف المنتج وجمع متطلبات العميل. ويمكن إنجاز ذلك من خلال عمل نماذج أولية للمنتج وتحليل العمل في سيناريوهات صغيرة (storyboarding) (Song [et al.], 2005). لاكتشاف المتطلبات ذات العلاقة بالهيكلية الأساسية، وتفاعلاتها مع غيرها من متطلبات المشروع ووضع استراتيجيات للتصدي لأي

تعارض ناتج، نستخدم تقنية تعرف بالتحليل الشامل (Hofmeister [et al.], 2000).

بالنسبة إلى مشاريع البرمجة الموزَّعة في عدة مواقع، توفّر هندسة المتطلبات التي يحكمها نموذج البرمجة منافع عديدة تساعد في التغلب على بعض الأمور المتعلقة بالتواصل والتنسيق والتحكم. وتساعد النمذجة المرئية على التواصل والتفاهم بين فرق العمل الموزَّعة. وتوفر الأداة التي تستخدم لتقويم النماذج قياسات حسب الطلب لحجم المتطلبات والتقدم في المشروع ودرجة الثبات ومقدار ما تمَّ إكماله ودرجة الجودة. وعلى نحو أكثر أهمية، يمكن أن توفّر نماذج لغة النمذجة الموحدة دعماً لعمليات التتبع المؤتمتة؛ كما يمكن تتبع المتطلبات المنمذجة قُدماً ومطابقتها بالتصميم والاختبارات أو بشكل رجعي لمطابقتها بمتطلبات السوق وطلبات المساهمين وأهداف العمل. قد يساعد ذلك في التحقق من المتطلبات المشمولة (أي المتطلبات التي تم تنفيذها في إصدار معين) وتحليل تأثيرها (أي ما هو تأثير التغيير في المتطلبات).

ينصبّ تركيز خطوة تصميم هيكلية النظام على الحصول على فهم تفصيلي للمتطلبات الهيكلية المهمة وإنشاء هيكلية قابلة للتنفيذ. وتتعامل العديد من منهجيات البرمجة مع الهيكلية بطريقة غير مباشرة أو ضمنية. وتعتمد جودة النظم المبرمجة باستخدام إحدى منهجيات البرمجة هذه على مستوى مهارة وخبرة المخطِّط بشكل كبير. نستخدم طرقاً للهيكلة المركزية (Bass [et al.], 2003) كورش عمل خصائص الجودة (QAW) والتصميم المحكوم بالخصائص (ADD) وطريقة التحليل الهيكلي المتبادل (ATAM). وتوفر هذه الطرق توجيهاً منهجياً وصريحاً للمصمم لإنشاء النظم بالجودة المطلوبة. وتتعامل طريقة ورش عمل خصائص الجودة مع المساهمين مبكراً في دورة

حياة برمجة النظام لاكتشاف متطلبات الجودة التي تقود بدورها هيكلية النظام. أما طريقة التصميم المحكوم بالخصائص فتستخدم هذه المتطلبات لتصميم هيكلية النظام؛ بينما تساعد طريقة التحليل الهيكلي المتبادل المساهمين على فهم التسلسل في المشروع الذي يلي الهيكلية التي يتم اعتمادها.

يَستخدم مدير المشروع وثيقة خصائص المتطلبات وخصائص الهيكلية والنموذج لعمل خطة المشروع لتنفيذ المنتج الجديد (Paulish, 2002). ينفذ مدير المشروع تحليلاً للمخاطر ويقوم بتعريف استراتيجيات المشروع؛ كما يقوم بإعداد خطط للإصدار التزايدي(*) (incremental release) وخطة البرمجة المقترحة والتي يصف فيها كيفية برمجة المنتج وزمن ذلك. وتستخدم طرق التقويم لتقرير الجهد اللازم لبرمجة المنتج والجدولة الزمنية لذلك.

في خطوة برمجة المنتج، يتم تصميم نماذج البرنامج وتنفيذها والاختبار ودمجها مع إصدارات المنتج المتوسطة والتي قد يكون بعضها في موقع العميل للاختبار كإصدار بيتا (beta release). في آخر الأمر، يتم إنتاج خط أساس للمنتج ناضج بما فيه الكفاية للنشر في بيئة المستخدمين.

يتم تنظيم برمجة المنتج للحصول على الحلول المثلى باستخدام فرق برمجة صغيرة موزَّعة ومتزامنة بواسطة تنظيم مركزي. وتستلم فرق البرمجة الموزَّعة في مناطق مختلفة في العالم خصائص النموذج من الفريق المركزي. ويكون كل فريق مسؤولاً عن تصميم وتنفيذ واختبار النموذج المصمم؛ هذا ويقود كل فريق مدير التزويد

(*) إصدارات متكررة قصيرة تتكون كل منها من الإصدار السابق مضافاً إليه خصائص جديدة.

(supplier manager). في تجربتنا، يكون دور المدير المورّد غير محدَّد أو منفَّذ بشكلٍ كافٍ لكنه دور رئيس في منهجيتنا (انظر الفصل العاشر).

تعمل فرق البرمجة على تنفيذ فترات برمجية تكرارية متزايدة للنماذج باستخدام أسلوب الاختبار المقدّم آخذين بالحسبان أفضل الممارسات التي توفرها منهجيات البرمجة السريعة. وينفذ الاختبار لتجربة الواجهات البينية للنموذج الفريق المسؤول عن هيكلية النظام، وهذه الاختبارات لا تساعد في التحقق من النماذج وحسب، بل تساعد المبرمجين على فهم دلالات الواجهات البينية بوضوح (Simons, 2002). تصمم النماذج باستخدام عيِّنات تصميم معروفة ويتم مراجعتها بصورة مستمرة للحفاظ على جودتها وصيانتها (Fowler, 2004).

2-4 مراحل البرمجة والقرارات

نستخدم لتخطيط المنتج عملية ذات مراحل. وهي كالعملية الموحدة العلائقية القياسية (RUP Rational Unified Process) تتكون من أربع مراحل في دورة حياة العملية البرمجية (Arlow and Neustadt, 2002; Larman, 2005). المرحلة التحضيرية (inception phase) هي مرحلة النقطة الحرجة الوهمية التي يتم خلالها الاستقصاء عن الحلول المحتملة لتقرير الجدوى من المشروع. ما إن يتم الإعلان عن أن المشروع مجدٍ، حتى يدخل المشروع مرحلة التطوير (elaboration phase). وهي المرحلة ذات المعلم الهيكلي الرئيس التي يتم فيها تحديد أولويات المتطلبات وتقرير تلك الأولويات التي يجب تنفيذها أولاً لأهميتها للهيكلية. وفي نهاية مرحلة التطوير، يتم وضع خطة برمجة موثوقة حيِّز التنفيذ وتبدأ عندئذ مرحلة التنفيذ

(construction phase). هذه المرحلة هي مرحلة معلم الإمكانيات التشغيلية لأنه في نهايتها يتم توزيع البرمجية في بيئة الإنتاج (production environment) في موقع أحد العملاء أو أكثر كإصدار بيتا لشرح إمكانياته التشغيلية. حالما يتم تقرير أن البرمجية جاهزة من حيث القدرة التشغيلية، تنتقل إلى المرحلة الانتقالية (transition phase) وهي مرحلة حرجة لإصدار المنتج حيث يكون المنتج جاهزاً للتسويق.

فكرة مفيدة:

إجعل أعضاء الفريق البعيدين جزءاً من الفريق المركزي في المراحل الأولى من المشروع. تتطلب عدد من المتطلبات والأنشطة المرتبطة بالهيكلية تعاوناً لنقل الخبرات الفنية والهيكلية إلى الفرق البعيدة. من المستحسن إشراك أعضاء من الفرق البعيدة في مرحلتي التحضير والتطوير وذلك ليكتسبوا فهماً لمدى المشروع وللرؤية الهيكلية. ومن ثم يعود هؤلاء الأعضاء إلى مواقعهم حين بدء مرحلة التنفيذ ويتصرفون كخبراء محليين. إضافةً إلى ذلك، نقترح أن ينتقل هؤلاء الأعضاء الرئيسون من فرق العمل البعيدة إلى الموقع المركزي مع عائلاتهم. وهذا من شأنه التقليل من مشكلة الحنين إلى الوطن التي قد يعانيها الأعضاء وتوفّر دعماً محلياً حين ظهور مشكلة «الصدمة الثقافية».

تمتد هذه المراحل خلال الفترات البرمجية (iterations) العديدة، وتؤدي كل فترة برمجية إلى إصدار جزء مكتمل من المنتج قابل للتنفيذ. يعتمد عدد الفترات البرمجية وطولها في المراحل على

حجم المشروع. نستخدم تقنيات(*) (scrum) ;(Schwaber, 2004 ;Schwaber and Beedle, 2001)، وبناءً على ذلك يتم تقسيم الفترات البرمجية في المشاريع الكبيرة إلى فترات شهرية ذات فترات زمنية أقصر (sprints). يتم تخصيص رزم العمل للفرق لكل فترة برمجية والتي يتم إكمالها بطريقة متزايدة في حدود الشهر. تسمى هذه الفترات المتزايدة بالإصدارات الهندسية (engineering releases). في نهاية الفترة البرمجية، تشكِّل الإصدارات الهندسية العديدة إصداراً قابلاً للتنفيذ كجزء مكتمل من المنتج. أما في المشاريع الأصغر، قد تكون مدة الفترة البرمجية هي نفس مدة الفترة الشهرية، لذا فقد لا يكون هذا النوع من التخطيط ضرورياً.

تفصل بين هذه المراحل فترات اتخاذ قرارات إدارية. يتم مراجعة فترة اتخاذ القرار لتقويم نتائج المرحلة الحالية وتقويم الخطط المقترحة وتفويض الموارد التي يتطلب استثمارها للمرحلة التالية.

تستخدم عمليات التخطيط البرمجي ذي المراحل لتحقيق المنافع الآتية:

- عرض تطور المنتج كبؤرة ابتكار لأفكار جديدة. ويتم التحقق من أفكار عديدة، لكن يتم استثمار عدد صغير من أفضل الأفكار لتصبح منتجاً أو جزءاً من منتج.
- تتطلب المراحل المبكرة استخدام مصادر أقل من المراحل اللاحقة حيث يتطلب الأمر توظيف عدد أكبر من فرق البرمجة لتنفيذ واختبار المنتج الجديد.

(*) إطار عمل تزايدي تكراري لإدارة العمل المعقد (كبرمجة منتج جديد) وغالباً ما تستخدم هذه التقنية في تطوير البرمجيات السريعة.

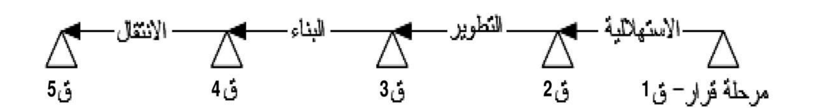
- يتم توجيه التواصل بين المبرمجين وموظفي المبيعات في الميدان في مراحل مبكرة. لا يشجع موظفو المبيعات على بيع الأفكار الجديدة أو الأفكار التي سيتم برمجتها وتطويرها في زمن قريب بدلاً من بيع الأفكار والمنتجات المنفَّذة. وقد تزيد الأفكار المقترحة للمنتج من توقعات العملاء إذا ما تم الإفصاح عنها مبكراً، وقد يؤثِّر ذلك سلباً في مبيعات المنتجات الحالية.
- يتم تحديد أهداف الجودة المرجوة بصورة أفضل، إضافة إلى قياسها والسيطرة عليها. قد تستخدم مراجعات فترات اتخاذ القرار كبوابات يمكن للمنتج من خلالها أن يتواءم مع معايير جودة معينة قبل البدء في المرحلة التالية من عملية البرمجة.
- تمكّن مراجعات فترات اتخاذ القرار الإدارة من عرض الميزانيات المقترحة والتحكم بها لكل مشروع ولكل مرحلة من مراحله. فإذا ما كان أحد المنتجات واعداً من الناحية التسويقية ولكن برمجته تحتاج إلى استثمارات غير متوقعة، يتم تعديل ميزانية السنة المالية أو يتم التبادل بين المشاريع.
- يمكن إيقاف المشاريع غير الواعدة. تساعد المراحل وفترات اتخاذ القرار الإدارة على تقرير كيفية الاستمرار الأفضل في المشاريع المستقبلية وموازنة المحافظ المالية للمنتجات.

قد تتكون عملية تطوير منتج جديد من بعض التحسينات المضافة على المنتج الحالي ويتم تسويقها كإصدار جديد أو نسخة معدلة جديدة أو منتج جديد ضمن قطاع سوق موجود، أو كخط إنتاج جديد. يركز هذا الكتاب على المبادرات الجديدة حيث يكون هناك هيكلية لبرمجية جديدة أو معدلة لتنفيذ المنتج. وبناءً على ذلك، لن تذكر الحالات التي يتم فيها إضافة خصائص جديدة ثانوية لمنتج متوفر وله هيكلية. في هذه الحالة، تقوم الشركات عادة بإعداد خطة

إصدار تزايدية، ويتم عمل التحسينات إلى أن يتم إطلاق النسخة الجديدة من المنتج إلى الميدان. يمكن أن تستخدم المنهجيات المقسمة إلى مراحل لإضافة الوظائف الكبرى للمنتج حيث يتم الاستثمار فيها بالتوازي مع تعريف متطلبات الإصدارات المستقبلية التالية للمنتج الذي يتم تطويره حالياً.

نشدّد في هذا الكتاب على المرحلة المبكرة من مراحل عملية تطوير منتج برمجي. فإذا ما تمت عمليات تحليل المتطلبات وتصميم هيكلية النظام ووضع خطة المشروع بصورة جيدة خلال المراحل الأولى من المشروع، فمن المحتمل أن تتم برمجة المنتج حسب نطاق المشروع (scope) والجدول الزمني والميزانية المخصصة ما لو تم تخطي هذه الأنشطة أو تنفيذها تنفيذاً هزيلاً. ومن ناحية أخرى، لا يتم إكمال هذه الأنشطة بصورة تامة أبداً، وغالباً ما تُترك بعض الأمور غير المكتملة إلى المراحل الأخيرة من عملية البرمجة. وعليه يتم توفير مجموعة من المبادئ التوجيهية والتوجيهات العامة والنصائح لإدارة المراحل المبكرة من عملية تطوير المنتج البرمجي بكفاءة.

يوضح الشكل 2-2 مثالاً على عملية تخطيط منتج ذات مراحل. أما أنواع القرارات الإدارية (ق) في المثال، فهي:



الشكل 2-2: مثال على عملية تخطيط المنتج ذات المراحل

- ق1: قرار بدء البحث عن (منتج أو تقنية أو سوق) للمنتج الجديد أو المنتجات الجديدة.

- ق2: قرار بدء عملية تحديد المتطلبات وتحليلها وبدء تصميم هيكلية النظام المتقدم.
- ق3: قرار بدء البرمجة. يتم مراجعة خطة المشروع المقترحة في هذا الاجتماع لتحديد مجال العمل والجدول الزمني والاستثمار المطلوب لبناء المنتج حسب المتطلبات والهيكلية المعروفة ضمن مرحلة التطوير.
- ق4: قرار إطلاق المنتج في السوق لاستعمال العملاء. وقد تضيف بعض الشركات مرحلة قرار متوسطة يتم خلالها الإعلان عن المنتج في الميدان قبل الانتهاء من برمجته.
- ق5: قرار الأفول أو إخراج المنتج من السوق. وقد تضيف بعض الشركات مرحلة قرار متوسطة يتم خلالها تقرير متى سيتم التوقف عن بيع ذلك المنتج لعملاء جدد، ومتى سيتم التوقف عن صيانة المنتج المتوفر لدى بعض العملاء.

2-5 الملخص والاستنتاجات

حددنا في هذا الفصل الأمور المتعلقة بمشاريع تطوير البرمجيات الموزَّعة المراكز وعرض العوامل التي تعتبر حاسمة لنجاح هذا النوع من المشاريع. كما تم شرح إطار العمل لتطوير البرمجيات التي يعمل فيها فرق عمل موزَّعة التي تفعل عوامل النجاح الحاسمة في عنونة بعض القضايا المرتبطة بمشاريع تطوير البرمجيات الموزَّعة المراكز. إضافة إلى شرح إطار العمل الإداري لتقويم المرحلة الحالية وبدء المرحلة التالية.

2-6 أسئلة للمناقشة

1. أذكر بعض الفروق المهمة بين مشاريع تطوير البرمجيات المنتظمة في موقع واحد والمشاريع الموزَّعة في عدة مناطق.

أذكر بعض الاستراتيجيات المستخدمة لإدارة هذه الفروق.

2. برِّر سبب كون هندسة المتطلبات المحكومة بمنهجية البرمجة متفوقة على هندسة المتطلبات القائمة على النصوص. لماذا تعتقد أنه من المهم بصفة خاصة توزيع مشاريع تطوير البرمجيات؟
3. ما أهمية أساليب الهيكلية المركزية لهيكلية النظام ولوضع خطة المشروع؟
4. تعتبر مراحل القرارات المختلفة في بداية ونهاية مراحل تطوير البرمجية بمثابة بوابات الجودة. ماذا تفهم من هذه العبارة؟

المراجع

Books

Arlow, Jim and Ila Neustadt. *UML and the Unified Process: Practical Object-Oriented Analysis and Design*. Boston, MA: Addison-Wesley, 2002.

Bass, Len, Paul Clements and Rick Kazman. *Software Architecture in Practice*. Second Edition. Boston, MA: Addison-Wesley, 2003.

Hofmeister, Christine, Robert Nord and Dilip Soni. *Applied Software Architecture*. Boston: Addison-Wesley, 2000.

Larman, Craig. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. Third Edition. Upper Saddle River, N. J.: Prentice Hall, 2005.

Paulish, Daniel J. *Architecture-Centric Software Project Management*. Boston: Addison-Wesley, 2002.

Schwaber, Ken. *Agile Project Management with Scrum*. Redmond, Wash.: Microsoft Press, 2004.

———— and Mike Beedle. *Agile Software Development with Scrum*. 1st Edition. Upper Saddle River, N. J.: Prentice Hall, 2001.

Periodicals

Herbsleb, James D. and Audris Mockus. «An Empirical Study of Speed and Communication in Globally Distributed Software Development.» *IEEE Transactions on Software Engineering*: vol. 29, Issue 6, June 2003, pp. 481-494.

Conferences

Berenbach, Brian. «The Automated Extraction of Requirements from UML Models.» *Proceedings of the 11th Annual IEEE International Requirements Engineering Conference (RE'03)*, Monterey Bay, CA, 8-12 September 2003, pp. 287-288.

———. «The Evaluation of Large, Complex UML Analysis and Design Models.» *Proceedings of the 26th International Conference on Software Engineering (ICSE 2004)*, Edinburgh, Scotland, U.K., 23-28 May 2004b, pp. 232-241.

———. «Towards a Unified Model for Requirements Engineering.» *Fourth International Workshop on Adoption-Centric Software Engineering (ACSE 2004)*, Edinburgh, Scotland, U.K., 23-28 May 2004a, pp. 26-29

Song, Xiping [et al.]. «S-RaP: A Concurrent, Evolutionary Software Prototyping Process.» *Proceedings of the Software Process Workshop*, Beijing, China, 25-27 May 2005.

Websites

Fowler, Martin. «Using an Agile Software Process with Offshore Development.» April 2004, Retrieved on 6 November 2005, from Martin Fowler Website: <http:// www. Martinfowler.-com>.

Simons, Matt. «Internationally Agile.» *InformIT*: 15 March 2002, Retrieved on 6 November 2005, from InformIT Website: <http:// www. informit. com>.

القسم الثاني

التخطيط



الفصل الثالث

هندسة متطلبات النظام

وجد ألبيرتو وهو مدير مشروع (BAS)، أن مشروعه في مأزق أليم؛ حيث كان أعضاء فريق الاختبار يحصلون على جزيئات من البرمجية شهرياً لاختبارها من دون معرفة الوظائف التي تم تنفيذها؛ ولم يكن واضحاً بالنسبة إليه (أو لأي شخص آخر) ما الذي ستسلّمه فرق البرمجة في الفترات البرمجية التكرارية، وكان المشروع يبتعد أكثر وأكثر عن الجدول الزمني المقرر.

علاوةً على ذلك، أصبح زميله سوبو، الذي يعمل في فريق هندسة المتطلبات، عاجزاً بوجود عدد كبير من المتطلبات الحالية الموجودة في قاعدة بيانات كاليبر (caliber database)، بل إنه كان مغرقاً بعدد كبير من المتطلبات الجديدة التي تم إضافتها. ولأن عملية برمجة المنتج مستمرة منذ ست سنوات ولا يبدو في الأفق نهاية لها، كان من الصعب تتبع كيفية تغيّر تصميم النظام مع الوقت ليلائم جميع المتطلبات الجديدة الناشئة.

إن السبب الجذري للمأزق الذي وقع فيه كل من ألبيرتو وسوبو

هو تجمع عدد كبير من الأنشطة التي انحرفت عن مسارها. وفي ظل غياب عملية مناسبة لهندسة المتطلبات لمشروع تطوير البرمجيات الموزَّعة المراكز، سيكون من الصعب تجنُّب مثل هذا الوضع. وسنركِّز في هذا الفصل وفي الفصل التالي على هندسة المتطلبات بشكل أو بآخر.

نركز في هذا الفصل على عملية هندسة البرمجيات الوظيفية والقضايا المرتبطة بتطوير البرمجيات الموزَّعة المراكز ومنهجيتنا في التعامل مع هذه القضايا. أما في ما يتعلق بدعم عامل النجاح الحاسم «الحد من الغموض والالتباس»، فيركِّز نهجنا تركيزاً كبيراً على توضيح خصائص المتطلبات وتوفير تتبع لتحديد مدى تغطية المتطلبات وتأثير التغيير الذي يطرأ عليها. ويُناقش في الفصل الرابع كيفية تعريف وإدارة المتطلبات المهمة بالنسبة إلى الهيكلية وكيفية ربطها بالمتطلبات الوظيفية وتضمينها في سياق مشروع تطوير البرمجيات الموزَّعة المراكز.

3-1 تمهيد

كما هو الحال بالنسبة إلى الجوانب الأخرى لتطوير البرمجيات، فإن الأمور المتعلقة بهندسة المتطلبات التي قد تعاني منها مشاريع تطوير البرمجيات الموزَّعة المراكز قد توجد في المشاريع المنتظمة أيضاً. ولكن احتمالية وتأثير هذه الأمور لا يكون نفسه في مشاريع تطوير البرمجيات الموزَّعة المراكز والمشاريع المنتظمة. فعلى سبيل المثال، قد تكون إدارة التغيير معضلةً صعبة الحل في جميع المشاريع؛ ففي مشاريع تطوير البرمجيات الموزَّعة المراكز من الصعب إجراء تحليل دقيق لتأثير التغيير في المتطلبات بسبب الرؤية المحدودة للأنشطة التفصيلية لفرق البرمجة الموزَّعة في عدة مناطق. وللسبب ذاته، يصعب أيضاً ضمان جودة كافية للبرمجة (code) التي يجب تسليمها.

وكما هو الحال في جميع المشاريع، تعتمد الأنشطة الأخرى (كعمليات التصميم والاختبار والتخطيط) على عملية هندسة المتطلبات. وهذا يُلقي عبئاً إضافياً على جهود هندسة المتطلبات لأن العمليات أكثر تعقيداً في مشاريع تطوير البرمجيات الموزَّعة المراكز، وذلك لتنفيذ هذه الأنشطة بالطريقة الملائمة. ويُناقش في هذا القسم هذه القضايا بمزيد من التفصيل، ثم يُتابع في الفصل شرح المنهجية المتبعة للتعامل مع هذه المشكلات.

3-1-1 إدارة التغيير

إدارة التغيير هي من أكثر الأمور صعوبة حتى في أفضل الظروف. نتحدث في هندسة التغيير (على الأقل في هذا السياق) عن إدارة التأثير المضاعف للتغيير في المتطلبات. تتضمن عملية إدارة التغيير ما يلي:

- **تحليل التأثير**: ما هي تكلفة التغيير؟ نحتاج الحصول على فهم دقيق للأنشطة التي ستتأثر بالتغيير (والمهام المسندة لفرق العمل).
- **تحديث الأعمال المرتبطة**: يجب تحديث المتطلبات والتصاميم المرتبطة.
- **إعادة التخطيط**: إذا تم قبول التغيير، يلزم تعديل الجدول الزمني للمشروع.

كما ذُكر سابقاً، من الصعب الحصول على تفاصيل واضحة للأنشطة التي تقوم بها فرق العمل العديدة في مشاريع تطوير البرمجيات الموزَّعة المراكز. ففي ظل غياب التتبع الواضح بين المتطلبات والتصميم، قد يكون تحديد الفرق المتأثرة بتغيير المتطلبات أمراً صعباً. في مشاريع تطوير البرمجيات الموزَّعة المراكز، لا تكون الاجتماعات التي تعقد لوضع التصاميم ومراجعة المشروع سهلة أو

ذات كفاءة نظراً إلى عدم توفر أعضاء فريق العمل في موقع واحد.

هذا ويجب وجود عملية تتبع واضحة بين المتطلبات وخطة المشروع. في بيئة العمل السريعة التقليدية، تكون العلاقة بين المتطلبات والخطة ديناميكية. فعندما نقوم بتعديل وضبط الخطة بانتظام، نحصل على تتبع واضح بين المتطلبات والخطة لتسهيل عملية تحليل التأثير وإعادة التخطيط. سنناقش كيفية عمل ذلك في القسم الآتي.

3-1-2 ضمان الجودة

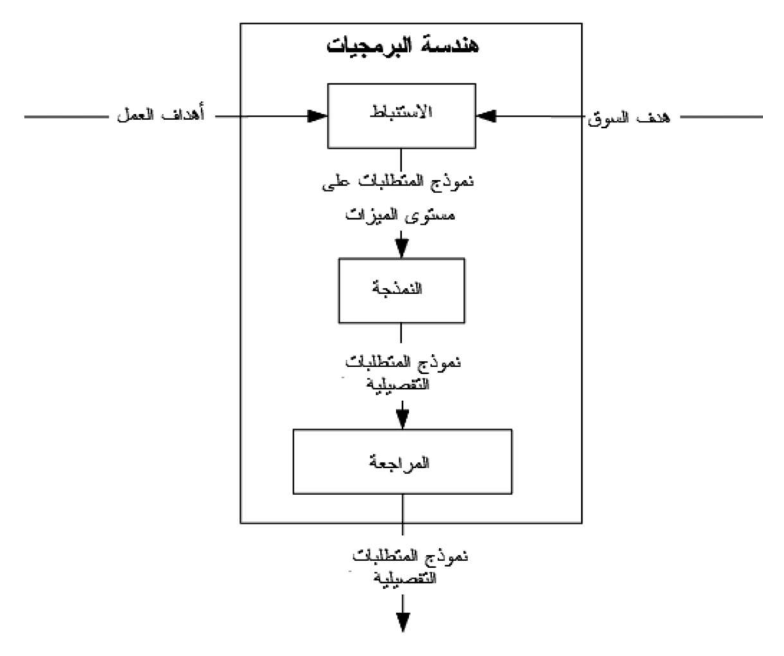
إن الأنشطة المتعلقة بضمان الجودة مهمة جداً في مشاريع تطوير البرمجيات الموزَّعة المراكز. ومن المهم الحصول على ردود الأفعال والانطباعات الدقيقة عن تطور عمل فرق العمل المختلفة وعن المشروع كاملاً بصورة دورية. ومن الأمور التي قد تكون صعبة في العديد من المشاريع وجود خطط للاختبار ذات تغطية كافية يتم تنفيذها بالتوازي (أو تقريباً بالتوازي) مع عملية بناء النظام. تدعو المنهجيات البرمجية السريعة (agile approaches) إلى أن ينفذ المبرمجون اختبارات (unit test) كجزء من عمليات البرمجة (مثال: كما في منهجية البرمجة التي تحكمها الاختبارات). ندعو إلى الالتزام بهذا الإجراء أيضاً، لكن المنهجيات البرمجية السريعة لا تعالج قضايا التكامل أو اختبار النظام المتقاطع بين الفرق. لقد وجدنا أن هذا أمر خطير في مشاريع تطوير البرمجيات الموزَّعة المراكز. فلتتمكن من تنفيذ مستوى معيَّن من الاختبار، يجب تحديد متطلبات النظام بتفاصيل كافية بحيث تكون المخرجات التي يجب تسليمها واضحة ومتوقعة من فرق البرمجة في كل فترة برمجية. يساعد هذا في ضمان أن المبرمجين متفقون ومنتجون ويحصلون على الدعم اللازم ليكونوا بالتالي قادرين على بناء النظام.

3-1-3 أثر هندسة المتطلبات في العمليات المترابطة

حتى نكون قادرين على تنفيذ عمليات التصميم والتخطيط والاختبار بطريقة ملائمة في مشاريع تطوير البرمجيات الموزَّعة المراكز، نحتاج إلى عملية هندسة المتطلبات التي تدعم هذه الجهود. بمعنى آخر، يجب تتبع مخرجات هذه العمليات؛ يجب أن ينفَّذ أي تغيير يحصل على إحدى هذه المخرجات بسهولة على أساس أنه تعديل على المخرجات الأخرى. في حالة ألبيرتو المعروضة سابقاً، لو توفرت هذه العملية لحلت معظم مشاكله. قد يعني ذلك المزيد من التشدد (كما هو متوقع في منهجية البرمجة الانحدارية waterfall)، غير أن ذلك غير صحيح. وسنناقش هذا بمزيد من التفصيل في الفصل المخصص لعملية وضع خطة المشروع (انظر الفصل السابع من هذا الكتاب).

3-2 عملية هندسة المتطلبات

تم التطرق في القسم السابق إلى أهمية المتابعة في مشاريع تطوير البرمجيات الموزَّعة المراكز. وتعتمد منهجيتنا في الحفاظ على المتابعة على إنشاء نماذج للمتطلبات والتصميم في لغة النمذجة الموحدة. إضافة إلى ذلك، نقوم بتحليل المتطلبات بتسلسل هرمي بدءاً من الخصائص الوظيفية (features) إلى المتطلبات التفصيلية لتسهيل المطابقة بين المتطلبات وخطة المشروع. كما نعمل على تكوين حالات الاختبار (test cases) من نموذج المتطلبات مباشرة (انظر الشكل 3-1).



الشكل 3-1: مخطط عمل لهندسة المتطلبات

نشرح في هذا القسم منهجيتنا لتحديد المتطلبات وتوثيقها وتحليلها ومراجعتها. ويتضمن ذلك تحديد الأنشطة والمشاركين والمدخلات والمخرجات المرتبطة بالعملية. كما نناقش أيضاً كيفية تنفيذ هذه العملية أثناء مراحل دورة حياة البرمجية.

3-2-1 تحديد المتطلبات

قبل بدء هذا النشاط، يجب تحديد رؤية المنتج أو هدف السوق. يجب أن يكون هناك تعريف منطقي لسبب قيام الشركة بإنتاج هذا المنتج، والغرض الذي سيُستخدم المنتج من أجله، وما هو السوق المستهدف. إن عدم توفر خارطة طريق صلبة تعرّف قطاعات السوق المستهدفة (فقد يكون مفاجئاً وجود فكرة عامة واحدة فقط

عن الأسواق والمناطق المستهدفة)، سيكون من الصعب عملياً تحديد المستخدمين والعناصر التي سيتم التغاضي عنها، وسيتم وضع الافتراضات، وسيتم بذل الكثير من الجهد في المجالات غير المرغوب بها أو المطلوبة.

نقوم بإجراء عملية تحديد المتطلبات أثناء الاجتماعات المجدوَلة التي تهدف إلى استخلاص المعرفة الميدانية من الخبراء وتعريف الخصائص والميزات التي يتضمنها منظور المشروع وتلك التي لا يتضمنها، وتوثيق الميزات في نموذج ذي منطقية وتنظيم متقدم.

إن المُخرج الأساسي لعملية تحديد المتطلبات هو نموذج للمتطلبات على مستوى الخصائص. لا نستهلك الوقت أثناء اجتماعات تحديد المتطلبات في مناقشة النموذج بالتفصيل وإكماله، بل نتأكد من أننا نحدد أهم الخصائص ونقوم بتنظيم النموذج بحيث يتم إكمال التفاصيل في ما بعد. وعندما تصبح هذه النماذج أكثر تعقيداً، يصبح التنظيم أكثر أهمية. وكما هو الحال في التصميم المعقَّد، من غير المألوف إجراء إعادة العمل (refactor) وإعادة التفكير في تنظيم النموذج نفسه لتعزيز المتوالية المنطقية أثناء إنجاز أنشطة تحديد المتطلبات. إن استغراق الوقت والتفكير في تنظيم المتطلبات نفسها سيوفر الوقت على المدى الطويل، إذ سيكون نموذج التنظيم الضعيف غير عملي وصعب الاستخدام.

3-2-1-1 المشاركون

يوضح الجدول 3-1 المشاركين في عملية تحديد متطلبات المشروع ومسؤولياتهم.

فكرة مفيدة:

إجعل قادة فرق العمل التي تعمل عن بُعد يشاركون في

عملية تحديد المتطلبات. عادة ما يستغرق الأمر وقتاً لجعل الفرق التي تعمل عن بُعد تسرع في عملها. ويدخل في ذلك الوقت المستغرق في نقل المعرفة في مجال العمل. تصبح عملية نقل المعرفة أكثر سهولة حين حضور مندوبين عن فرق العمل التي تعمل عن بُعد في اجتماعات تحديد المتطلبات. وقد يساعد ذلك في تطوير السياق واللغة المشتركة التي يتقاسمها الجميع وتفادي الالتباس لاحقاً.

الجدول 3-1: المشاركون في عملية تحديد المتطلبات

الوظيفة	المسؤولية
مهندس المتطلبات (requirements engineer)	تحديد المتطلبات في النموذج.
المعاون (facilitator)	تنسيق اجتماعات تحديد المتطلبات والتأكد من استمرارية وتيرة واتجاه الاجتماع لا تزال مثمرة. كما يقوم المعاون بتسجيل الأسئلة والقضايا والمتطلبات المرتبطة بالهيكلية والأمور الأخرى على اللوح.
خبراء مجال العمل	يمثل هؤلاء الخبراء دور المستخدمين الذين سيعملون على النظام ويشرحون الخصائص والتفاصيل عن كيفية استخدام النظام.
مديرو المشروع	يقوم مديرو المشروع بتحديد منظور المشروع ومجاله. وهم غالباً ما يضطلعون بمسؤوليات في المشروع بعد تحديد خصائص المشروع المتقدمة.
المصممون	المصمم يمثل فريق تصميم هيكلية المشروع، ويجب أن يساهم في أنشطة هندسة المتطلبات. أما الاحتمال الآخر فهي وجود عضو أو عضوين من فريق هندسة المتطلبات في فريق التصميم.

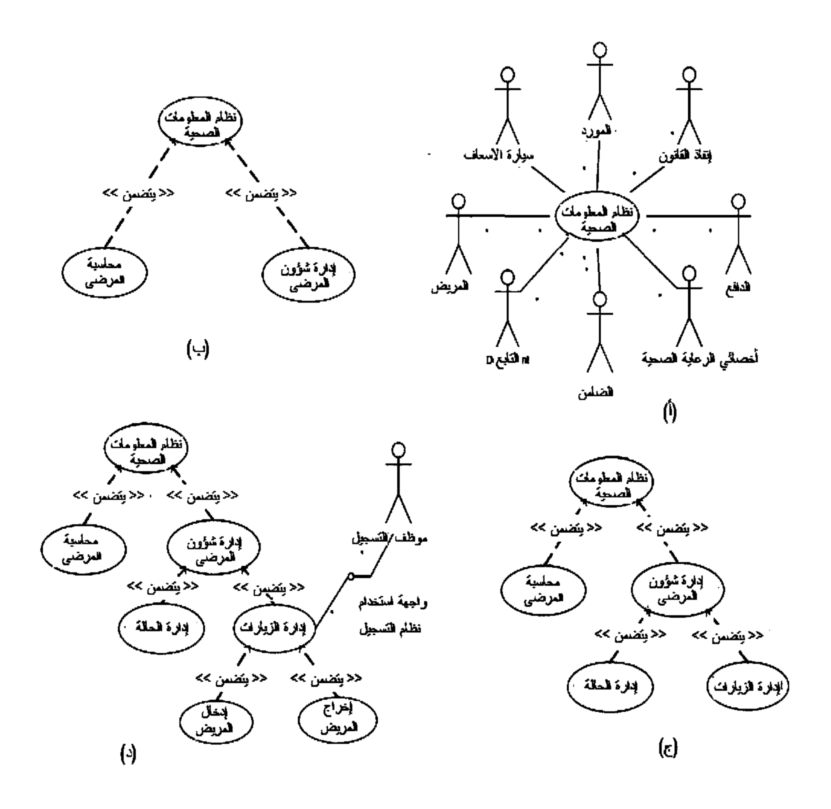
3-2-2 النمذجة

نوضح في هذا القسم التنظيم الموصى به لنموذج المتطلبات الذي يوجِّه القضايا التي تم تحديدها في الأقسام الافتتاحية من هذا الفصل. نقوم بنمذجة المتطلبات بلغة النمذجة الموحدة (Booch [et al.], 2005; Fowler, 2004). ففي حين قد يتحمس بعضهم لاستخدام لغة النمذجة الموحدة أو قد يعارضونه (لمبررات منطقية)، وجدنا أن هذه منهجية فعَّالة نظراً إلى مستوى دعم الأدوات والقدرة على إدراج الوصف النصي والمزج بين التنظيم والمرونة. ونقوم أساساً بنمذجة المتطلبات كمجموعة من حالات الاستخدام التي تم تحليلها من النظام الأساسي والمتطلبات التفصيلية الخاصة بالنظم الفرعية (Berenbach, 2003, 2004a). وتتكون عملية تحليل المستوى الأول من الخصائص والمزايا ذات المستوى الأعلى التي تتوزع إلى خصائص ومزايا فرعية (استخلاص حالات الاستخدام). وفي نهاية المطاف، يتم تحليل الخصائص إلى حالات استخدام (حالات الاستخدام الواقعية). وقد وجدنا أن هذه المنهجية توفر العديد من الفوائد (Berenbach, 2004b)، منها:

- أن نموذج لغة النمذجة الموحدة أسهل من النص الحر من حيث التصفح والفهم.
- هذا النموذج هو بمثابة خطة عمل مشتركة للاتصال بين المشاركين في المشروع.
- يمكن التحقق آلياً من الاتساق والاكتمال والتحليل لضمان الجودة.
- يمكن استخدامه لإنشاء متطلبات النظام والاختبار وخطط المشروع بطريقة شبه آلية.

تبدأ عملية النمذجة بمخطط بياني لسياق النظام، يبين النظام

وكافة الجهات الخارجية التي ترتبط به. يظهر النظام كحالة استخدام نظرية ويُستخدم بمثابة نقطة واحدة للإدخال في نموذج المتطلبات. إن حالة الاستخدام النظرية هي صقل تدريجي لإنتاج حالات الاستخدام الواقعية. يُظهر الشكل 3-2 عملية النمذجة لنظام معلومات صحية (HIS) تمَّت برمجته لشبكة الصحة التكاملية (IHN).

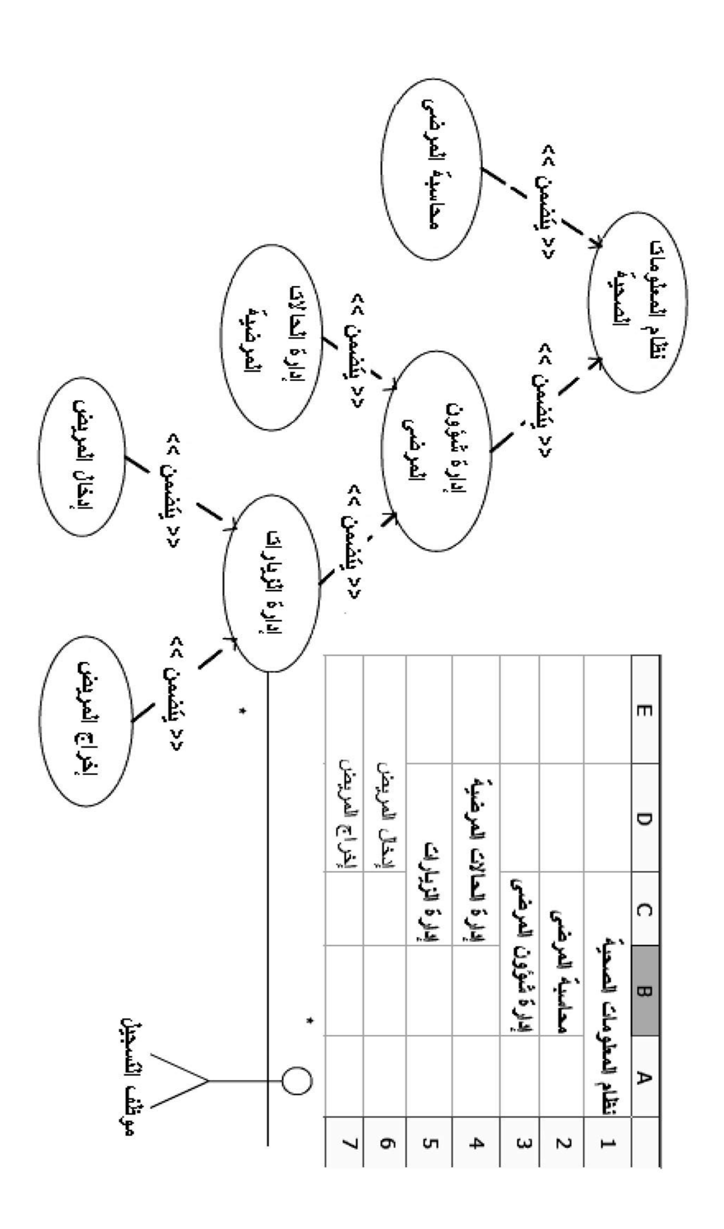


الشكل 3-2 : متطلبات نظام المعلومات الصحية (أ) سياق النظام (ب) الخصائص الأساسية (ج) الخصائص الجزئية (د) حالات الاستخدام الواقعية

لقد تم تبسيط المخطط البياني عن قصد بسبب محدودية مساحة الصفحة. أما في الواقع، فسيكون هناك المئات من الخصائص والآلاف من حالات الاستخدام الواقعية. وبناءً على ذلك، سيكون هناك العديد من المستويات الهرمية.

إن حالات الاستخدام الواقعية هي خدمات يجب توفرها في نظام المعلومات الصحية، وهي بالتالي تمثل المتطلبات التفصيلية للنظام. يمكن استخلاص هذه المتطلبات آلياً من النموذج الموضح في الشكل 3-3 وهو بمثابة مدخل لخطة المشروع.

في نهاية عملية النمذجة (قد يزعم الكثيرون أن هذه العملية لا تنتهي أبداً)، سيظهر إلى حيِّز الوجود نموذجاً لمتطلبات النظام. نستخدم النموذج للحفاظ على عملية متابعة التصميم وخطة المشروع. وسيتم مناقشة ذلك بمزيد من التفصيل في فصل «عملية التخطيط» (انظر الفصل السابع من هذا الكتاب)؛ لكن عندما تقوم بتقويم المشروع ووضع خطته، عليك أن تكون قادراً على وضع قائمة بالوحدات اللازمة لمجموعة محددة من خصائص وميزات البرنامج. ويمكن إنجاز ذلك عن طريق رسم المتطلبات التفصيلية للميزة والبرامج المرتبطة بها والتي ستتحقق فيها جميع المتطلبات. إضافةً إلى ذلك، فإنك ستقوم بحزم الخصائص في إصدارات وتقسيم عمليات البرمجة على فترات قصيرة شهرية. كان بإمكان ألبرتو وسوبو التخفيف من مشاكلهما إلى حدٍ معقول لو أنهما استخدما هذه الطريقة لنمذجة المتطلبات وتتبعها.



الشكل 3-3: استخلاص متطلبات النظام من النموذج

فكرة مفيدة:
قم بإعداد نموذج للمتطلبات باستخدام طريقة «البحث التوسعي أولاً»(*) breadth first.
غالباً ما يكمن الغموض في التفاصيل. فإذا ما حاولت نمذجة المتطلبات باستخدام طريقة «البحث بطريقة التعمق أولاً»(**) فإنك ستستغرق المزيد من الوقت في مناقشة التفاصيل قبل أن يصبح النموذج ثابتاً ومستقراً. إضافة إلى ذلك فإن الأمر يتطلب بعض الوقت قبل تجهيز النموذج المتقدم كمادة إدخال للمصممين. من الأهمية بمكان وجود تبادل دائم بين فريقي إعداد الهيكلية وإعداد المتطلبات. إن عرض الخصائص المتقدمة مبكراً لفرق إعداد الهيكلية مع التحسين المتعاقب يساعد على تسهيل عملية التبادل هذه.

3-2-2-1 المشاركون

تبدأ عملية النمذجة أثناء اجتماعات تحديد المتطلبات (الجدول 3-2). أثناء ذلك، يتم وضع خصائص النظام في نموذج وتحليلها إلى خصائص فرعية. يتم التركيز على تنظيم هذه المتطلبات بطرق صحيحة وتحديد المشاركين الذين سيتعاملون بها. يتم لاحقاً تفصيل المتطلبات

(*) طريقة متبعة في نظرية المخططات (graph theory)، وهي خوارزمية للبحث، تبدأ من العقدة الجذرية ثم تنتقل إلى العقد المجاورة. ثم لكل واحد من تلك العقد الأكثر قرباً، تستكشف فروعها المجاورة غير المستكشفة، وهلم جراً إلى أن يتم الوصول إلى الهدف، والشكل 3-3 مثال على ذلك.

(**) هي خوارزمية للعبور أو البحث في هيكلية شجرية أو رسم بياني، تبدأ العملية في عقدة الجذر (حيث يتم اختيار عقدة كجذر في حالة الرسم البياني) ثم تتحرى البحث عن طول كل فرع إلى أبعد عقدة ممكنة قبل التراجع.

عن طريق تحليل الخصائص الفرعية إلى حالات استخدام واقعية وإضافة مخططات الحالة والنشاط والتفاعلات ما بينها. يقوم مهندس المتطلبات بإعداد النماذج أثناء تنفيذ عمليات التحليل. وغالباً ما يوجد خبير في مجال العمل المطلوب تنفيذه (على الرغم من أنه قد تكون هناك أوقات يعمل فيها مهندس المتطلبات على وضع مسودة لمفاهيمهم، وذلك بهدف مراجعتها مع الخبير لاحقاً). هذه القائمة التي تتضمن الأشخاص هي قائمة أقصى قائمة يمكن وضعها، لكنها ستتغير بناءً على نوع العمل المحدد الذي يتم إنجازه ومستوى المعرفة التي يتطلب من مهندس المتطلبات اكتسابها وبنية فريق العمل.

الجدول 3-2: المشاركون في عملية نمذجة متطلبات النظام

الوظيفة	المسؤولية
مهندس المتطلبات (requirements engineer)	تحديد المتطلبات في النموذج.
خبراء مجال العمل (domain experts)	يمثلون وجهة نظر المستخدمين ويصفون خصائص النظام وتفصيل كيفية استخدامه.
مديرو المنتج (product managers)	يقوم مديرو المنتج بتعريف مدى المشروع. عادة ما يضطلعون بمهام في المشروع عند تعريف الخصائص المتقدمة.
المصمم	شخص ينوب عن فريق تصميم هيكلية النظام يشارك في عملية هندسة المتطلبات. الاحتمالية الأخرى هو أن يكون أحد مهندسي النظام (أو أكثر) عضواً في فريق تصميم هيكلية النظام.

3-2-3 مراجعة متطلبات النظام

قد يصبح نموذج متطلبات النظام غير عملي أحياناً ويصعب فهمه، وقد يظهر فيه تعارض في المتطلبات. لهذه الأسباب، نقوم

بمراجعة المتطلبات وتحليلها وإعادتها نظراً إلى توفُّر الإدراك المشترك والنماذج القابلة للاستخدام. أصبح استخدام الأدوات أمراً حاسماً لإدارة التغيير في المتطلبات. فعندما تقوم بإعادة صياغة النموذج وإعادة تكوين المتطلبات، فإنك لا ترغب في إضعاف خطة المشروع أو أن تخفِّض من درجة تتبع التصميم وإلا تبعثرت العملية.

تمكننا الطريقة التي نستخدمها لإنشاء نموذج المتطلبات من عمل بعض التحليل (التركيبي) آلياً (Berenbach, 2004b). يمكننا التحقق من إكمال النموذج والتأكد من عدم انتهاك القواعد العديدة (مثلاً، الروابط التكرارية)، إلى غير ذلك. يجب أن ينفِّذ التحليل الدلالي يدوياً أشخاص لديهم معرفة بالمتطلبات ومجال العمل. نقوم بعقد اجتماعات مراجعة دورية لإنجاز النوع الأخير من التحليل. ويجب أن يلتقي في هذه الاجتماعات الأشخاص الذين اشتركوا في عملية تحليل المتطلبات إضافة إلى المساهمين الذين لم يشاركوا بشكل مباشر. ولاجتماعات المراجعة هذه ثلاثة أهداف هي: (1) ضمان فهم المتطلبات وتفسيرها في النموذج بالشكل الصحيح، (2) التحقق من أن نموذج المتطلبات مفهوم ومنطقي ومن أن الذين يحتاجون لفهمه يستطيعون قراءته بسهولة، (3) وإجراء المزيد من التحليل للمتطلبات. وقد وجدنا أن أفضل طريقة لفهم ومعرفة المتطلبات هي المشاركة بطريقة ما في الاجتماعات الخاصة بمتطلبات النظام. ففي حين أن مشاركة الجميع في أنشطة هندسة المتطلبات أمر غير عملي، غير أنه من المفيد مشاركة خبراء مجال العمل والمصممين ومصممي هيكلية النظام ومديري التزويد في عملية مراجعة المتطلبات.

3-2-3-1 المشاركون

يبين الجدول 3-3 أعضاء فريق العمل والمساهمين الذين يشاركون في مراجعة متطلبات النظام. تشير خبراتنا إلى أنه يمكن

مراجعة أربعة إلى عشرة متطلبات في الساعة أثناء اجتماعات مراجعة متطلبات النظام.

الجدول 3-3: المشاركون في عملية مراجعة متطلبات النظام

الوظيفة	المسؤولية
مهندسو المتطلبات	تحديد المتطلبات في النموذج.
خبراء مجال العمل	يمثلون وجهة نظر المستخدمين ويصفون خصائص النظام وتفصيل كيفية استخدامه.
مديرو المنتج	يقوم مديرو المنتج بتعريف مدى المشروع. عادة ما يضطلعون بمهام في المشروع عند تعريف الخصائص المتقدمة.
مصممو هيكلية النظام (architects)	شخص ينوب عن فريق تصميم هيكلية النظام يشارك في عملية هندسة المتطلبات. تكمن أهمية وجود المصمم في الاجتماع في أن المتطلبات تصبح مألوفة له، كما إنه يستطيع تحديد المتطلبات التي ستؤثر في هيكلية النظام.
مصممو النظام (system designers)	يجب أن يدرك المصممون الذين يقومون بإعداد التصميم التفصيلي ماهية المتطلبات بشكل أساسي. وجلسات مراجعة المتطلبات طريقة جيدة تساعد المصممين على تسهيل فهم وإدراك المتطلبات.
مديرو التزويد	وهم الخط الدفاعي الأول لفرق البرمجة. يجب أن يتفق المديرون مع بقية الأفراد العاملين في المشروع في ما يخص المتطلبات وهيكلية النظام والتصميم التفصيلي. إن اشتراك مديري التزويد في اجتماعات مراجعة متطلبات النظام طريقة جيدة ليسهل فهمهم لها.

3-3 الأدوات

من المهم استخدام بنية تحتية للأدوات مع الأخذ بعين الاعتبار مدى صعوبة الحفاظ على الاتساق بين نموذج المتطلبات والتصميم وخطة المشروع أثناء عملية إعادة صياغة نموذج ضخم معقد. نواجه وقتاً عصيباً في وصف البنى التحتية المحددة في ظل عالم الأدوات دائم التغير (وهذا خارج مجال هذا الكتاب). ومع ذلك، يتوجب ضمان وجود بعض الطرق الأوتوماتيكية لـ:

- إعداد توثيق مقروء ومفهوم من نموذج المتطلبات.
- تنفيذ تتبع ذي اتجاهين بين قاعدة بيانات المتطلبات والنموذج.
- تنفيذ التتبع بين نموذج المتطلبات وخطة المشروع (غالباً ما يكون ذلك من خلال التصميم).

إضافة إلى ما تقدم، ننصح بشدة عمل نماذج بدائية للبنى التحتية المقترحة إذا لم يكن ذلك متوفراً أصلاً. ولم نختبر إلى الآن أداة تتكامل فيها الجهود ليكون أداؤها على النحو المعلن عنه.

3-4 المرحَلة

هناك ثمة تكامل بين العمليات الفرعية التي تحدث بصورة تكرارية وقد تكون معقدة. تفرض عملية تحديد مراحل دورة حياة البرمجية قيوداً محددة على مخرجات الأنشطة بناءً على موقعك في دورة حياة النظام.

كما ناقشنا سابقاً (وستتم مناقشته لاحقاً) في هذا الكتاب، فإننا معنيون بالدرجة الأولى في المرحلة التحضيرية بوضع مفهوم للعمليات المتقدمة والخروج بمحاولات متقدمة وجدولة التقويم الزمني للمشروع ورسم نهج للطرق المنوي استخدامها والقضايا ذات العلاقة بالمشروع.

أثناء المرحلة التالية، مرحلة التطوير، يبدأ إعداد التصميم التفصيلي ووضع النماذج البدائية للنظام؛ كما سيتم تفصيل متطلبات النظام، ووضع الخطط التفصيلية للمشروع. وأخيراً، في مرحلة التنفيذ، قد يستمر تفصيل المتطلبات، ولكن تطرأ بعض التغييرات، ما يتطلب دمجها وتكاملها مع المتطلبات الأساسية؛ كما تتم في هذه المرحلة بناء وتكامل واختبار النظام؛ وتستمر عملية التخطيط ومراقبة التطور في المشروع.

خلال مراحل دورة حياة النظام جميعها، يتم تنفيذ جميع الأنشطة المشروحة مسبقاً. فهي تنفذ بمستويات عديدة من التفاصيل. وفي المرحلة التحضيرية تكون السمات الأساسية التي سيدعمها النظام هي الحاجة الأساسية للتخطيط وتقدير الفترة الزمنية والتصميم. وهذا يعني أنه يجب تحديد متطلبات النظام على مستوى الخصائص؛ هذا ويتم تحديد مجال ومدة برمجة النظام في هذه المرحلة، كما يتم تحديد أولويات خصائص النظام بطريقة أو بأخرى. يجب إجراء بعض التطوير والتفصيل لدعم عملية نقل المتطلبات ومطابقتها في هيكلية النظام.

بما إن التصميم يصبح أكثر تفصيلاً في مرحلة التطوير، يتم تحديد فترات سريعة ويتم توزيع المهام في هذه الفترات، وتصبح عمليات المتطلبات تفصيلية أكثر من ذي قبل. يجب أن يتم النقل المتقدم للمتطلبات في وحدات النظام على مستوى واجهة الاستخدام (على الأقل للفترات السريعة الأولى)، وهذا يعني أن المتطلبات يجب أن تكون مفصلّة بما فيه الكفاية للسماح بنقل المتطلبات إلى وحدات النظام. يجب تحديد تدفق البيانات قدر الإمكان. كما ينبغي تحديد المتطلبات المتخصصة (التي يتطلبها العميل لتلائم احتياجاته)

ومختلف أنواع التباين بين المتطلبات المتخصصة والمتطلبات الأساسية في هذه المرحلة. ويجب أن يتم تنسيق هذه الأنشطة مع خطط المشروع وأنشطة التصميم لضمان توفر التفاصيل حين احتياجها. وهذا يعني أن هذه الأنشطة تستمر أثناء مرحلتي التطوير والبناء حتى يتسنى لمهندسي المتطلبات فهم المتطلبات التي تحتاج إلى تطوير تفصيلي أولاً. وهذا يتأتّى من خطة المشروع ومصممي هيكلية النظام. وقد يحتاج مصممو هيكلية النظام المزيد من التفاصيل لاتخاذ قرارات التصميم المهمة. كما إن خطة المشروع ستحدد ترتيب إصدار الخصائص والميزات. ويجب أن يستخدم مهندسو المتطلبات ذلك لمساعدتهم على تخطيط أنشطتهم وترتيب أولوياتها.

3-5 الملخص والاستنتاجات

لقد وضّحنا في هذا الفصل عملية هندسة المتطلبات الوظيفية للنظام وكيفية تعاملها مع الأمور المرتبطة بتطوير البرمجيات الموزَّعة المراكز. وعلى وجه الخصوص، تُعتبر إدارة التغيير وضمان الجودة وتأثير هندسة المتطلبات في عمليات التصميم والتخطيط والاختبار من الأمور التي تحقق قدراً كبيراً من الأهمية في بيئة البرمجة الموزَّعة. ويتطلب الأمر استخدام عملية هندسة المتطلبات لإدارة هذه الأنشطة بكفاءة. ويتم إنجاز ذلك في المنهجية التي نتبعها من خلال نماذج لغة النمذجة الموحدة للمتطلبات وتتبع التصميم وخطط المشروع وعمليات الاختبار بوضوح. كما تسهّل منهجيتنا طريقة التعامل مع التغييرات التي تطرأ على متطلبات النظام. عادة ما يحدث التغيير نتيجة الانطباعات والآراء التي نحصل عليها من التفاعل مع العملاء والمساهمين أثناء الفترات البرمجية التكرارية العديدة. وقد يكون استيعاب التغيرات سهلاً في البيئة المنتظمة، لكن تنسيق المتغيرات في سياق تطوير البرمجيات الموزَّعة المراكز يكون أكثر

صعوبة نظراً إلى تأثيره المحتمل على فرق العمل الموزَّعة العديدة. إن الحفاظ على عملية المتابعة من الخصائص المتقدمة إلى المتطلبات الواقعية والتصاميم والاختبارات المرتبطة بها وخطط المشروع تساعد في إدارة التغيير في المتطلبات وتأثيرها في فرق العمل الموزَّعة.

6-3 أسئلة للمناقشة

1. ما هي مظاهر المشروع التي تتأثر بتغيّر المتطلبات خلاف هندسة المتطلبات؟
2. كيف تحدد أنشطة هندسة المتطلبات المشروحة في هذا الفصل القضايا التي تفرضها المتطلبات المتناسبة مع احتياجات مشروع تطوير البرمجيات الموزَّعة المراكز؟
3. ما هي العوائق التي تعترض تنفيذ مثل هذه العملية في شركة ما :

المراجع

Books

Booch, Grady, James Rumbaugh and Ivar Jacobson. *The Unified Modeling Language User Guide*. Second Edition. Boston, MA: Addison-Wesley, 2005.

Fowler, Martin. *UML Distilled*. Third Edition. Boston, MA: Addison-Wesley, 2004.

Conferences

Berenbach, Brian. «The Automated Extraction of Requirements from UML Models.» *Proceedings of the 11th Annual IEEE International Requirements Engineering* Conference (RE'03), Monterey Bay, CA, 8-12 September 2003, pp. 287-288.

———. «The Evaluation of Large, Complex UML Analysis and Design Models.» *Proceedings of the 26th International Conference on Software Engineering(ICSE 2004)*, Edinburgh, Scotland, U. K., 23-28 May 2004b, pp. 232-241.

———. «Towards a Unified Model for Requirements Engineering.» *Fourth International Workshop on Adoption-Centric Software Engineering (ACSE* 2004), Edinburgh, Scotland, U.K., 23-28 May 2004a, pp. 26-29.



الفصـــــل الرابع

المتطلبات اللازمة لهيكلية النظام

كان باولو يعاني التعب. فقد عاد من رحلة حول العالم لزيارة عدد من مواقع التطوير والبرمجة في محاولة منه لإعادة المشروع إلى مساره. فقد كانت عملية البرمجة تسير بصعوبة. وكانوا متأخرين عن الجدول الزمني للمشروع. وكانت البرمجة غير مستقرة. ولم يكن باولو يستسيغ ذلك كله. فقد كان النظام الذي كانوا يعملون على بنائه شبيهاً بالنظم الأخرى التي بنتها الشركة من قبل. وقد كانت عناصر النظام مجمَّعة بطريقة مرنة وواضحة المعالم إلى حد ما. ولا شك أن هذا المشروع افتقد إلى العقل المفكر. فقد كان مقتنعاً أن المشكلة كانت في فرق البرمجة التي تعمل عن بُعد؛ فقد بدوا كأنهم لم يفهموه قطّ.

في العديد من المشاريع، سواء كانت الموزَّعة أو المنتظمة، من المهم فهم وإدراك المتطلبات بصورة كافية وهذا أمر مهم. إذ ترتبط المتطلبات بهيكلية النظام في مراحله المبكرة بحيث تنعكس على قرارات التصميم الأساسية. أما إذا لم يتم إدراكها وفهمها، فقد

لا يتمكن النظام من اكتساب الخصائص والميزات المهمة إذ إنها ترتبط بالأداء وقابلية القياس وغير ذلك من مزايا النظم. هذا ولا تُكتشف المشاكل في هذه الجوانب عادة حتى مراحل متأخرة من دورة حياة النظام، عندئذ تكون صعبة التغيير حتى في المشاريع المنتظمة. وبالنسبة إلى مشاريع تطوير البرمجيات الموزَّعة المراكز، قد تقود هذه القضايا إلى توقف المشروع بسهولة. ويتضمن التعامل مع هذه الشؤون تنسيقاً كبيراً بين الفرق المسؤولة عن العديد من جوانب النظام، وهذا أمر يصعب تحقيقه في مشاريع تطوير البرمجيات الموزَّعة المراكز. وبفضل أحد عوامل النجاح الحاسمة «الحد الأقصى من الاستقرار» تعلمنا عبر الطريق الصعب أنه من المهم جداً جعل هذه المخاوف واضحة بطريقة تمكِّن من أخذها بعين الاعتبار في هيكلية النظام.

نناقش في هذا الفصل منهجيتنا في تحديد المتطلبات المحددة ونقلها إلى الفرق بطريقة منظمة بشكل عام، إضافة إلى مناقشة الأمور المرتبطة بمشاريع تطوير البرمجيات الموسعة. ثم نناقش في هذا الفصل الأنشطة التي تتضمنها عملية جذب ونقل المتطلبات المنظمة المهمة.

بعد قراءة الفصل السابق، قد يبدو عدم تخصيصنا فصلاً آخر لهندسة المتطلبات أمراً غريباً. فقد وجدنا أن هناك تصنيفين للمتطلبات التي يستخدمها مختلف المشاركون لأغراض مختلفة. وتستخدم المتطلبات الموضحة في الفصل الثالث لتصميم وظائف النظام. فهي لا تحدد التركيبة الضخمة للنظام. وبتنحية تقنيات تصميم الكوائن الموجهة، يمكن أن تقدم العديد من بُنى النظام المختلفة الوظائف المطلوبة. عملياً، يكون الحال غالباً أن يتم برمجة هيكلية النظام بالتوازي مع برمجة المتطلبات. ما الذي يستخدمه مصممو هيكلية

النظام لمساعدتهم في اتخاذ القرارات؟ وكيف يعرفون الخيارات التقنية الحاسمة وكيف يقيمون الخيارات؟ نتخذ منهجية برمجة مجموعة من الناقلات الهيكلية المرتبطة (أو متطلبات الهيكلية المهمة) لتحفيز تصميم الهيكل الإجمالي للنظام.

4-1 تمهيد

يجب أن يدعم شيءٌ ما هيكلية النظام (أو الهيكل الإجمالي للنظام البرمجي)، لكن ما هو هذا الشيء؟ هل هو متطلبات النظام الوظيفية؟ حسناً. .. نعم، لكن إذا دعمت العديد من الهيكليات المختلفة الوظائف بالتساوي تكون أفضل هيكلية يمكن استخدامها هي الأرخص، أليس كذلك؟ ليس بالضرورة، فإن ذلك يعتمد على السياق. وفي بعض الحالات، يكون الحصول على أسبقيَّة التسويق أمراً أكثر أهمية من الجودة والوظائف وإمكانية الصيانة. وفي حالات أخرى، قد تُستخدم الهيكلية لدعم خط الإنتاج وتكون تكلفة تنفيذ البناء المبدئي أكبر بمرتين أو ثلاث مرات من تكلفة بناء نظام مفرد. بمعنى آخر، يجب أن تدعم الهيكلية أهداف العمل الذي تقوم به الشركة. ونشرح في هذا القسم من الكتاب كيفية ترجمة أهداف العمل في هيكلية النظام بمزيد من التفصيل والمعلومات التي يتطلبها المصمم لدعم التصميم وكيفية دخول مشروع تطوير البرمجيات الموزَّعة المراكز في هذه المعادلة.

4-1-1 كيف ترتبط هيكلية النظام بأهداف العمل؟

ما هي العوامل التي تؤثِّر عملياً في قرارات التصميم بالنسبة إلى كل شخص ذي علاقة بتصميم النظام؟ كم مرة تفكر بكيفية تأثير القرارات التقنية المهمة في قدرة الشركة على تحقيق أهدافها؟ في العديد من المشاريع، من المحتمل أنك لا تناقش مثل هذه الأمور.

وعملياً، يتم العديد من النقاشات حول التصميم من دون وجود مجموعة واضحة من المعايير المتَّبَعَة في اتخاذ القرارات، مثل «هل نستخدم طريقة اتصال تعتمد على الأحداث أم على الرسائل؟» أو «هل نستخدم لغة البرمجة «NET» أو «J2EE»؟ تتأثر هذه المناظرات غالباً بالتفضيلات والآراء الشخصية والعوامل الأخرى التي يصعب قياسها كمياً أكثر من تأثرها بمعايير العمل. هذا، وقد تؤثر هذه الخيارات في خطوط الإنتاج في الشركة. نحن على علم بحالة واحدة على الأقل حدثت في شركتنا حيث نتج من اختيارات هيكلية البرمجية المهمة خسارة حصتنا في السوق. وفي نهاية المطاف، قمنا ببيع عدد كبير من وحدات العمل.

مثال على ذلك:

تحقق الشركة «A» جميع منتجاتها من خلال بيع أجهزتها ومعداتها. وتنتج الشركة تطبيقاً برمجياً يعمل على إدارة الشبكات لنظم المعدات، ولكن يوجد في الوقت الحالي خسارة (أي إن التطبيق يخسر مالاً لكنه يتيح إمكانية بيع العتاد) في حصة السوق، وفي بيع عدد كبير من وحدات العمل. يدرك المديرون التنفيذيون في الشركة أن المعدات أصبحت سلعة، ويتوقعون أن هوامش مبيعات المعدات ستضعف خلال العشر سنوات القادمة. ولضمان استمرارية أعمالهم لأطول فترة ممكنة، قرروا بناء نظام إداري جديد يمكن تحويله إلى أرباح. وقد تمَّت طريقتهم في تصور كيفية تحول هذه البرمجية إلى عمل مربح بما يلي:

1. الحد من تكاليف البرمجة الداخلية.

2. توسيع نطاق السوق.

تخطط الشركة «A» للحد من تكاليف البرمجة الداخلية عن

طريق استبدال التطبيقات الموجودة حالياً بتطبيق جديد. وترغب الشركة في توسيع نطاق السوق عن طريق الدخول إلى أسواق جغرافية جديدة وعن طريق فتح قنوات للمبيعات على شكل «موزعين ذوي قيمة مضافة value-added resellers». ويقوم هؤلاء الموزعون ببيع البرمجية باستخدام علاماتهم التجارية الخاصة لدعم نظم المعدات الخاصة إلى العديد من مختلف المصنّعين.

وبالنظر إلى المثال السابق، يمكننا أن نرى العديد من الجوانب التي يبدو فيها تأثير أهداف الشركة في هيكلية النظام تأثيراً واقعياً وأساسياً من دون التأثير في وظائفه. يجب اتخاذ بعض الاعتبارات لدعم خطوط إنتاج مختلف المعدات. وقد يوجد في مختلف الأسواق اعتبارات تنظيمية، وتؤثر فيها الثقافة السائدة في المنطقة (على سبيل المثال، في بعض المناطق، يكون العملاء مستعدين للدفع بسخاء مقابل الحصول على خدمات التقنيين المدربين لتركيب وصيانة المنتج، بينما لا يدفعون لهذه الخدمات في مناطق أخرى)، هذا بالإضافة إلى اعتبارات اللغة (على سبيل المثال، القراءة من الأعلى إلى الأسفل، أو ترجمة الرسائل التحذيرية).

وبالكشف عن المزيد من التفاصيل، تتحدد لدينا عوامل مؤثرة أخرى. إلى أيّ حد يمكن لأحد الحلول البرمجية دعم جميع هذه الأهداف؟ وما هي المقايضات والمخاطر التي تنطوي عليها؟ وهل يكون العمل مرتاحاً بوجود هذه المخاطر والمقايضات، أم هل يتوجب عليهم إدخال تحسينات وغربلة الأهداف (على سبيل المثال، الحد من الأسواق المنوي دخولها أو تغيير الجدول الزمني)؟ هذه جميعها هي قرارات عمل، لكنها تستلزم تدخل المختصين التقنيين. وغالباً ما يكون هناك فصل بين ما ترغب الشركة في تحقيقه وما يمكن برمجته وإنتاجه؛ وعليه تتسارع وتيرة المخاطر بغياب الجسر

الذي من شأنه التقليل من هذه الفجوة. تلاحظ هذه الحالة في مشاريع تطوير البرمجيات الموزَّعة المراكز. فمن الضروري جداً معالجة عدم التطابق المحتمل الموجود بين توقعات العمل وجدوى الحلول التقنية، نظراً إلى أن كل الأمور تكون أكثر تعقيداً وصعوبةً في مشاريع تطوير البرمجيات الموزَّعة المراكز.

4-1-2 ما هي العوامل المؤثرة في هيكلية النظام؟

كما أوضحنا في حالة باس (Bass [et al.], 2003)، تميل العوامل التي تؤثِّر في هيكلية النظام إلى أن تكون خصائص الجودة؛ فعلى سبيل المثال، الأداء والأمن وقابلية التعديل والموثوقية. ومن مثال أهداف العمل الموضح أعلاه، يمكننا أن نرى بعض المخاوف المتعلقة بقابلية التعديل (مثلاً، السماح للموزعين ذوي القيمة المضافة بإضافة العلامة التجارية الخاصة بهم إلى النظام) وتعديلات اللغة ومخاوف حول قابلية التكيّف (مثلاً، القدرة على تهيئة النظام في خط إنتاج المعدات). وقد تنبثق مخاوف أخرى بسهولة كالأداء والموثوقية والتوفر، وهي أمور غير واضحة في المثال. في مشاريع تطوير البرمجيات الموزَّعة المراكز، هناك بعض العوامل الإضافية يجب أخذها بالحسبان غير تلك المتعلقة بخصائص الجودة. وسنناقش هذه الأمور بتوسع في قسم لاحق.

غالباً ما ترى متطلبات تعرف بـ «غير وظيفية». فقد وجدنا أن ذلك قد يؤدي إلى نقاش أو جدل جانبي بطريقة أو بأخرى. وغالباً ما تتضمن المتطلبات المرتبطة بالهيكلية كلا النوعين وظيفية وغير وظيفية. فبدلاً من التركيز على إذا ما كان المتطلب «س» ذا طبيعة وظيفية أو غير وظيفية، نركز على إذا ما كان للمتطلب تأثير في البنية الإجمالية لهيكلية النظام. إن أحد المقاييس المعتمدة التي

يمكنك استخدامها لتحدد ذلك هو التساؤل عمّا إذا كان بمقدورك إنجاز المتطلب بكفاءة، بغضّ النظر عمّا إذا اتخذت القرارات بشأن الهيكلية الأولية ذلك بالحسبان. إذا كانت الإجابة «لا»، فإنه ينبغي على مصمم الهيكلية النظر في ذلك المتطلب في المراحل المبكِّرة من عملية التصميم.

4-1-3 ما هي المعلومات التي يحتاجها مصمم الهيكلية؟

لا يكفي القول (كما ورد في القسم السابق من هذا الكتاب) إن النظام يجب أن يكون قابلاً للتعديل والتكيّف ويسمح بتعديل اللغة، إذ إن أي نظام يكون قابلاً للتعديل من ناحية ما، كما يمكن أن يتم تعديل أي نظام في ما يتعلق بأي جانب بتوفر الوقت والمال الكافيين. والأسئلة التي تتبادر إلى الذهن هي: من أي ناحية يكون النظام قابلاً للتعديل، ومتى يكون ذلك، وكم يلزم من الوقت لإجراء التعديل؟ وما هي الجوانب المحددة في النظام التي يجب تعديلها؟ وهل تحتاج إلى التعديل أثناء التشغيل، أم أثناء عملية ترجمة شيفرة البرمجة، أم أثناء التفويض أو أثناء البرمجة؟ وهل يستغرق إتمام هذا التعديل ساعة من الزمن، أم أسبوعَي عمل، أم عشر سنوات؟ وتنطبق هذه الأسئلة على كافة الشؤون الأخرى. ويجب أن تكون التعديلات محددة بما يكفي لاختبارها في ما يتعلق بالهيكلية. فعلى سبيل المثال، قد يستطيع الشخص المسؤول عن المراجعة تحديد إذا ما كانت الهيكلية قادرة على إنجاز هذا المتطلب. يشرح باس (Bass [et al.], 2003) كيفية عمل ذلك بالتفصيل. ومن الأمثلة الأخرى في هذا السياق ما يلي:

- إذا فشل «المتحكم بالمحول» في الاستجابة أثناء الحمل الطبيعي، يستشعر النظام هذا الفشل ويتابع التواصل مع النظام خلال ثانيتين.

- سيكون المبرمجون قادرين على استبدال النظام القائم بذاته بآخر يعمل من خلال شبكة الإنترنت خلال شهريْ عمل.
- ستتم برمجة النظام بمستوى الجودة «ج»، ما يوفر مجموعة من وظائف تهيئة الجهاز المتصل (CDC) خلال اثني عشر شهراً تقويمياً باستخدام ما لا يزيد على 360 شخصاً.
- سيكون مهندس النظم قادراً على استيراد اللغات الغربية غير المدعومة مسبقاً إلى النظام أثناء التشغيل من دون تدني مستوى أداء النظام إلى ما دون مستوى متطلبات الحمل العادي أثناء فترة الكمون.

4-1-4 ما هو تأثير تطوير البرمجيات الموزَّعة المراكز في هيكلية النظام؟

يختص هذا الكتاب بتطوير البرمجيات باستخدام فرق العمل الموزَّعة في عدة مواقع. لماذا إذاً استغرقنا نصف الفصل في مناقشة متطلبات هيكلية النظام بينما تطرقنا إلى تطوير البرمجيات الموزَّعة المراكز بشكل أقل؟ هناك عدة أسباب تجعلنا نعتقد أن هذه المواضيع مرتبطة بمشاريع تطوير البرمجيات الموزَّعة المراكز على عدة مواقع.

أولاً، لا يُصرح الناس غالباً بالمتحكمات التي تقود هيكلية البرمجيات. ففي حين يكون ذلك مقبولاً عندما لا يختلف النظام قيد البرمجة عن آخر نظام تم تطويره بحيث يمتلك الأشخاص المعنيون الخبرة في مجال العمل، غير أنه قد يؤدي إلى تأخير المشروع «Gotchas». وتسببت هذه الاستكشافات غير المرغوب بها في تفكك العديد من المشاريع الموزَّعة في عدة مواقع؛ إذ إنها تتطلب إعادة بعض العمل غير المخطَّط له مسبقاً. وقد يطال ذلك العديد من جوانب النظام. ولن تتمكن فرق العمل من التنسيق بشكل كافٍ لتعديل الوضع في الوقت الملائم نظراً إلى طبيعة المشروع الموزَّعة.

ثانياً، حتى لو كانت هذه أمور يدرك الأشخاص المعنيون كيفية التعامل معها حدسياً،غير أنهم يجب أن يكونوا قادرين على جعل سياق ذلك الحدس صريحاً وواضحاً للفرق التي تعمل عن بُعد. ولا يمتلك أعضاء فرق العمل التي تعمل عن بُعد نفس الخبرة والبديهة، بل يحتاجون كذلك إلى المزيد من المساعدة لفهم الدوافع والأسس المنطقية لاتخاذ قرارات معينة.

غالباً ما يقال إن عملية تطوير البرمجيات الموزَّعة المراكز على عدة مواقع تتطلب هيكلية ذات مكوّنات «معرّفة جيداً ومجمّعة بطريقة غير محكمة». وفي مثل هذه الحالة، لا تكون هذه المعرفة كافية لضمان تصميم نظام يمكن تنفيذه برمجياً. فمن الطبيعي أن تكون خيارات المصممين مبنية على مهارات الأشخاص الذين يضطلعون بعمل في المشروع. هذا ويجب اتخاذ اعتبارات مشابهة في الحسبان في مشاريع تطوير البرمجيات الموزَّعة المراكز على عدة مواقع. غالباً ما يكون هذا الأمر معقداً، إذ قد يحضر بعض أعضاء فريق البرمجة من خارج الشركة، ما يعني جهلهم في مجال العمل. كما إنه من الشائع أن تكون المعرفة في مجال العمل المطلوب محصورة في قسم أو موقع معيَّن. ويجب وضع وحدات العمل في الاعتبار منذ بداية تصميم النظام. كما يجب تحديد فرق العمل اللازمة لوحدات العمل إذا كان ذلك متاحاً بحيث يمكن المواءمة بين الفريق والعمل الذي يجب إنجازه. إضافة إلى ما تقدم، تقترن وحدات العمل مع النظام بمستويات وطرق مختلفة. وبشكل عام، يتضمن الاقتران الأكثر إحكاماً وتعقيداً المزيد من التنسيق بين الفرق. إذا كنت تخطط لتوزيع العمل على فرقٍ غير قادرة على التنسيق والتواصل مع المجموعات المطلوبة، فلا بد أن يكون الوضع لديك غير ملائم.

تؤدي هذه الاعتبارات إلى تحليل يختلف عما قام المصممون بتصميمه. إضافة إلى ذلك، تكون الارتباطات بين المتطلبات مقيدة،

إذ يلزم تنفيذ بعضها مع متطلبات أخرى في الهيكلية. ويجب أخذ هذه المبادلات في الحسبان في ضوء أهداف المشروع. ولا تكون التغييرات التي تطرأ على الهيكلية الخيار الوحيد. فإذا كان هناك تعارض لا يوفر خيارات جذابة، فمن الممكن ضبط تنظيم الشركة بطرق تساعد على ضمان التطابق بين الخيارات التقنية وقدرات الشركة التطويرية (تم شرح ذلك في الفصل السادس من هذا الكتاب ـ إدارة المخاطر).

نعتقد أن مشاكل باولو نشأت عن عدم العناية بهذه العوامل على نحوٍ كافٍ. إذ لو تم إنشاء الهيكلية بما يتلاءم مع الهيكل التنظيمي للشركة وبالتنسيق بين الفرق التأسيسية لما عانى تلك الكوابيس المزعجة.

4-2 متطلبات الهيكلية المهمة

4-2-1 تحديد متطلبات النظام

غالباً ما نستخدم تقنية معيّنة لتحديد متطلبات الهيكلية المهمة (ASR) بناءً على منهجية ورشة عمل خصائص الجودة التي يعقدها بمعهد هندسة البرمجيات SEI (Bachmann [et al.], 2002; Barbacci [et al.], 2000). إن الهدف من هذا النشاط هو تأسيس مجموعة من أكثر متحكمات النظام أهمية مرتَّبة حسب الأولويات على هيئة سيناريوهات خصائص الجودة، التي تتمُّ مطابقتها مع أهداف العمل. ونقوم بعقد مراجعات دورية لتحديد المتطلبات حسبما يكون ذلك ضرورياً.

لضمان ممارسة العوامل التنظيمية التأثير الملائم في الهيكلية، قمنا بتطوير منهجية لجلب هذه المعلومات إلى المتطلبات المحددة

بالهيكلية (Hofmeister [et al.], 2000; Paulish, 2002). تهدف هذه المنهجية إلى فهم خصائص الشركة وتعريف المناطق والنطاقات المرتبطة بالمصممين وإيصال هذه الاعتبارات بطريقة تسمح للمصممين بالتعامل معها. هذا، وقد تم شرح هذه المنهجية في الفصل السادس من هذا الكتاب أيضاً.

قبل البدء بتنفيذ هذا النشاط، يجب توفر أهداف النظام التي غالباً ما تكون أهدافاً عامة. ويُفترض أن الجميع يعلمون سبب قيام الشركة بإعداد ذلك النظام. فإذا لم تتوفر خطة واقعية للأسواق المستهدفة واحتياجات هذه الأسواق والفروق بينها والاعتبارات الخاصة بكل منها، فالوقت ما زال إذاً مبكراً لعقد ورشة العمل هذه. وإذا لم يقم أحدهم بتوضيح هذه البنود بشكل تفصيلي واضح، يكون ذلك مؤشراً على وجود خطة عمل غير كافية للمشروع. إضافة إلى ذلك، يجب ضمان الرعاية لهذه العملية؛ ونظراً إلى طبيعة أصحاب المصالح المتنوعة، ويستلزم الأمر وجود شخص ذي مستوى إداري متقدم لضمان مشاركة أصحاب المصالح.

4-2-1-1 ورشة عمل حول متطلبات الهيكلية المهمة

لقد قمنا بتطوير ورشة عمل للمتطلبات المهمة من الناحية الهيكلية بناءً على ورشة عمل حول سمات الجودة عقدها معهد هندسة البرمجيات (Bachmann [et al.], 2002; Barbacci [et al.], 2000). وفي حين أن معظم الخطوات التي ستنفذ في ورشة العمل شبيهة بتلك التي نفذت في ورشة عمل سمات الجودة، قمنا بتصميم هذا النهج في بعض المجالات ليلائم أغراضنا بشكل أفضل. وفي ما يأتي عرض للخطوات الأساسية في ورشة عمل أهم المتطلبات من ناحية الهيكلية.

الخطوة 0: التحضير

تنفذ هذه الخطوة قبل عقد ورشة العمل. والهدف منها هو ضمان تحقق شروط المتطلبات السابقة للنجاح. ومن خبراتنا السابقة (والتي كان بعضها مؤلماً)، قمنا بصياغة هذه الخطوة لإتاحة الفرصة لإلغاء أو تأجيل ورشة العمل إذا لم تتحقق المتطلبات السابقة للنجاح. ويجب تحقق ثلاثة شروط لنجاح ورشة العمل (واضحة بجلاء):

- **ضمان المشاركة المناسبة لأصحاب المصلحة**

وهو غالباً الجانب الأكثر صعوبة في هذه الطريقة. وقد تكون المجموعة المثالية من أصحاب المصالح كبيرة نوعاً ما، وتشمل المبيعات والصيانة والجودة والخدمات والتواصل والتسويق وغير ذلك من المشاركين. ومن الضروري وجود راع أو كفيل من الإدارة العليا يتفهم أهداف ورشة العمل ويؤمن المالَ اللازم لها وذلك لضمان الحضور الملائم. ومن الضروري أن يوجد عدد من الحضور بالحد الأدنى:

- شخص يستطيع تصوير الأهداف الإجمالية للعمل المرتبطة بالمشروع أو المنتج (السبب وراء هذا المشروع).
- ممثلون عن العملاء والسوق يكونون قادرين على تحديد الأسواق الرئيسة ووصف احتياجاتهم وبيئاتهم (ما يتطلبونه لبيع المنتج بنجاح أو اعتبار المشروع ناجحاً).
- مصممو الهيكلية.

- **سياق العمل معلوم**

قد يبدو ذلك تعبيراً غير ضروري؛ فقد قمنا بإضافة هذا التعبير صراحةً بعد المشاركة في أحد المشاريع أو أكثر حيث لم يكن الهدف

من بناء المنتج غير واضح. وقد وفرنا في البداية مستوىً عالياً من التوجيه لما كنا نبحث عنه في سياق عرض السياق. ولكن بعد الحصول على جميع أشكال العروض قمنا بتطوير قالب أكثر تفصيلاً مزوداً بأمثلة لمساعدة الأشخاص في الحصول على المعلومات التي يطلبونها.

- **إجراء الترتيبات اللوجستية**

نظراً إلى صعوبة ضمان الأوقات الملائمة للحضور، نجد أنه من المحبط ضياع الوقت في البحث عن الأقلام الخاصة بالألواح وأجهزة العرض وغير ذلك من المعدات. ولتجنب هذا الوضع قمنا بإنشاء قائمة تدقيق لضمان إجراء كافة الترتيبات الملائمة.

الخطوة الأولى: عرض طريقة ورشة عمل متطلبات الهيكلية المهمة

إن الهدف من هذه الخطوة هو ضمان أن يفهم جميع الحضور المنهج المتَّبع وما الذي يتوقعونه في ورشة العمل هذه. وسيتم تزويد أصحاب المصلحة بمواد للقراءة المسبقة بحيث تصبح المنهجية مألوفة لهم، ولكننا وجدنا أنهم نادراً ما يقرأون أو يفهمون المنهجية من خلال مطالعة المواد المطبوعة التي نوفرها لهم. لذا نبدأ الاجتماع بشرح المنهجية واستقبال الاستفسارات وتوضيحها لهم. ونخصص ساعة ونصف لهذه الخطوة.

الخطوة الثانية: تقديم سياق العمل

نرغب في تكوين قائمة مختصرة بأهداف العمل المشتقة للمشروع على شكل نقاط في نهاية هذه الخطوة. ولعمل ذلك، نطلب من أصحاب المصالح تقديم ما يأتي:

- سياق العمل (مقدّمة عامة عن أهداف المشروع).
- الأسواق الرئيسة (وثيقة الصلة) والتي تصف:
  - العملاء المستهدفين
  - مدى أهمية هؤلاء العملاء (كأن يكونوا مصدراً رئيسياً للدخل، أو ألا يكونوا ذوي أهمية ربحية، ولكنهم يلزمون للحفاظ على السمعة، وغير ذلك).
  - جوانب المشروع التي قد تجعل هؤلاء العملاء يشترون المنتج.
- اعتبارات أخرى (كالبرنامج الذي يهدف إلى تقليل الزمن اللازم لإيصال المنتج إلى السوق لخطوط المنتج س، ص، ع بنسبة 25 بالمئة).

من النادر أن تكون كافة المعلومات متوفرة بالدقة والتفاصيل المطلوبين. ولضمان توفر تفاصيل تسمح بالاستفادة الفاعلة من ورشة العمل، نطلب من الشخص المسؤول تزويدنا بمسودات العروض التي سيتم تقديمها قبل بدء ورشة العمل.

الخطوة الثالثة: تحديد المتحكمات الرئيسة

الهدف من هذه الخطوة هو بدء ترجمة متطلبات العمل إلى متطلبات هيكلية واقعية. وهذه المتطلبات عامة بطبيعتها وتوفر سياقاً للخطوة التالية من خطوات إنشاء السيناريوهات. ومثال على ذلك، اعتبار إحدى النقاط المذكورة أعلاه.

إن بيع النظام الخاص بنا إلى الموزعين ذوي القيمة المضافة هو جزء من الاستراتيجية التي سنستخدمها لتوسيع السوق. ويجب أن يتمتع الموزعون ذوو القيمة المضافة بالقدرة على إضافة واجهات الاستخدام وإنجاز التهيئة والإعدادات الخاصة بهم أو القدرة على بيع

برمجياتهم كعلامات تجارية. وإضافة إلى ذلك، قد يبيعون معدات ذات صلة صنعها مورّد آخر.

بالنظر إلى ذلك، نستطيع أن نرى أن النظام يجب أن يكون قادراً على استيعاب واجهات الاستخدام والإعدادات المختلفة التي يتم إضافتها، ما يتيح للموزعين بيع المنتج بعلاماتهم التجارية الخاصة. إضافة إلى ذلك، سيتطلب النظام القدرة على التكيّف مع بروتوكولات عديدة تستخدمها أجهزة مضمِّنة أخرى بحيث يديرها النظام. هذه النقاط ليست واقعية وصلبة بما فيه الكفاية لاستخدامه كمتطلبات للهيكلية لكنها تعبيرات عامة عن الأمور التي يجب أن تكون الهيكلية قادرة على تحقيقها لدعم أهداف العمل المنصوص عليها أعلاه. خلال هذه المرحلة من مراحل ورشة العمل، نقوم بسرد «المتحكمات الرئيسة» ونستخدمها لتحديد السيناريو.

الخطوة الرابعة: سيناريوهات العصف الذهني(*)

الهدف من هذه الخطوة هو تحويل المتحكمات الرئيسة إلى واقع. ونقوم بذلك عن طريق الطلب من أصحاب المصلحة بتقديم سيناريوهات خصائص الجودة (Herbsleb and Grinter, 1999) التي تمثل اعتبارات معينة. قام معهد هندسة البرمجيات بعمل رائع في تحديد ماهية سيناريو خصائص الجودة بدقة، كما قاموا في المعهد بتصنيف مجموعة من السيناريوهات العامة لاستخدامها كدليل. ولن

(*) العصف الذهني (brainstorming) هو طريقة عملية لجلب عدة حلول لمشكلة معينة، بحيث يجتمع مجموعة من الأشخاص المعنيين لتحديد المشكلة ثم التفكير بحلول لهذه المشكلة بحيث يتم كتابة جميع الأفكار التي تتعلق بالمشكلة في ورقة ثم محاولة اختيار المناسب منها لحل المشكلة. وعادة ما تُستخدم عملية العصف الذهني في الحالات الآتية: حل المشاكل وبناء فرق العمل والإعلانات التجارية والتخطيط العملي وإدارة المشاريع.

نقوم بإعادة سرد تلك المعلومات هنا، غير أننا نقول إنه من المهم تحديد مقاييس مرتبطة لتحويل السيناريوهات إلى واقع. وبالنسبة إلى المتحكمات المذكورة أعلاه، قد يكون المقياس على شكل مستوى من الجهود اللازمة لإضافة واجهة استخدام جديدة أو مستوى الجهد اللازم لتطوير محوِّل بروتوكول، وهذا من الأمور التي يصعب عملها. فالأشخاص غير معتادين على التفكير في احتياجات العميل في ما يتعلق بهذه الأمور. فمن المستحيل فهم الهيكلية التي يجب إنجازها من دون بعض القياس الكمي. أما ما نقوم به غالباً عندما لا يكون أصحاب المصلحة قادرين على تحديد وقياس احتياجاتهم فهو الطلب منهم عمل تخمين مدروس والإشارة إلى أن هذا القياس ليس مؤكَّداً، ومن ثم الانتقال إلى المرحلة التالية. عملياً، يجب مراجعة هذه المقاييس عدة مرات أثناء عملية التصميم، وإنه لمن المساعد معرفة المقاييس المخمَّنة من تلك المقاييس الثابتة تماماً. وفي هذه المرحلة لا نقوم بأي جهد لتصفية السيناريوهات أو جمعها أو تحديد أولوياتها أو رفضها. ويُعتبر هذا جلسة عصف ذهني، ونتعامل معه على هذا الأساس. أما السيناريوهات المرتبطة فسيتم تحديدها في خطوات تالية. ولن يكون ذلك جهداً مضنياً. سيكون هناك العديد من المتغيرات على هذه السيناريوهات والتي لن يتم تحقيقها هنا، لكنك تحتاج إلى الحصول على مجموعة تمثِّل تلك المتغيرات. وغالباً ما نطلب من كل من أصحاب المصلحة عرض سيناريو واحد على الأقل.

الخطوة الخامسة: دمج السيناريوهات

حالما نحصل على السيناريوهات الأولية نقوم بتنقيتها وهذا لدمج السيناريوهات. ونقوم باختيار السيناريوهات التي يجب توحيدها وجمعها معاً، ومن ثم العمل على تحديد السيناريو الذي يشتمل على جوهر المجموعة. إن أساس هذه العملية هو ضمان أن الشخص

الذي عرض هذا السيناريو هو الذي له الكلمة الأخيرة حول إذا ما كان السيناريو الناتج يوصل الفكرة التي كان يعنيها عندما قدَّم العرض. وإذا لم يوافق ذلك الشخص على السيناريو، نقوم إما بتعديله إلى أن يحوز رضا ذلك الشخص أو نتركه خارج المجموعة.

الخطوة السادسة: تحديد أولويات السيناريوهات

ثمة طرق عديدة لتحديد أولويات السيناريوهات: يمكنك اتخاذ منهجية شاملة واستخدام بروتوكول تطبق فيه مجموعة المهتمين بالمشروع كاملة اقتراعاً على السيناريوهات التي يشعرون بأنها الأكثر أهمية؛ أو يمكنك الطلب من الأشخاص الذين يمثلون قطاعات العمل فقط تحديد أهمية السيناريوهات أو يمكنك استخدام طريقة أخرى لتحديد الأولويات. الأمر المهم هو أن تقوم بترتيبها على محورين وأن يكون لديك بروتوكول واضح ومنظَّم لترتيبها (وبخلاف ذلك، سينجم نقاش لا ينتهي). ويجب أن ترتب السيناريوهات من حيث الأهمية بالنسبة إلى أهداف العمل (يجب أن يكون ذلك بعمل مطابقة سهلة، إذ إن السيناريوهات معدة في سياق واحد أو أكثر من المتحكمات الرئيسة)، كما يجب ترتيبها حسب صعوبة إنجازها من وجهة النظر الهيكلية أو البرمجية. وهذا يقود إلى بعض الأمور المهمة للعمل والصعبة الإنجاز. هناك بعض المتحكمات (حوالى ثلاثة) التي ينتهي بها الأمر في تشكيل الهيكلية (قد يكون لهذه المتحكمات العديد من السيناريوهات التي تمثلها).

الخطوة السابعة: الختام وإيجاز النتائج

يتضمن النشاط الأخير تلخيص نتائج ورشة العمل بطريقة منظمة ومتماسكة وإخبار أصحاب المصلحة بما سيتم عمله بهذه النتائج وكيفية إكمال العملية. ومن الضروري تذكّر أن هذه النتائج لا تُعتبر نهائية. وتضمن ورشة العمل بشكل أساسي أن الجميع لديهم نفس

المفهوم للأهداف وتوجّه المشروع، وأنه تم تأسيس بنية معرّفة جيداً لمتطلبات الهيكلية وقناة اتصال ولغة مشتركة بين مصممي الهيكلية وأصحاب المصلحة. في هذه المرحلة يجب أن يتم عرض عملية استمرارية تصفية ودمج المتطلبات (غالباً ما نعقد اجتماعات للمراجعة من أصحاب المصلحة بصورة منتظمة مجدولة).

فكرة مفيدة:
لا تقم بتحديد أكثر من خمسة متحكمات هيكلية. عملياً، من الشائع وجود العديد من المتطلبات (حتى أهم المتطلبات من ناحية الهيكلية)، لكن يوجد غالباً القليل من الأمور التي تحكم الهيكلية حقاً. فعلياً هناك متحكَّمان إثنان أو ثلاثة من هذه المتحكمات. بينما يكون هناك العديد من المتطلبات التي تعوِّض المتحكمات، فإنها متنوعات عن نفس الشاغل. أما إذا زاد الأمر على ذلك بكثير تصبح الهيكلية معقدة جداً. إذا وجدت أن لديك العديد من الشواغل، جرب أن تجمعها في مواضيع مرتبطة متقاربة من بعضها بعضاً. فإذا ما زال لديك العديد من الشواغل، تأكد من ترتيب أولوياتها بوضوح بطريقة تجعل أمر تحديد أي منها يلزم جعل الهيكلية الأمثل لتحقيقه.

4-2-1-2 المشاركون

تتضمن الوظائف التي تم تحديدها للمشاركة في ورشة عمل متطلبات الهيكلية المهمة ما يأتي:

- **قائد ورشة العمل**: يكون هذا الشخص مسؤولاً عن قيادة وتيسير أمور ورشة العمل؛ ويجب أن يكون لديه:

- معرفة شاملة وكاملة بالأهداف وخطوات المنهجية.
- معرفة بهيكلية البرمجية.
- مهارات التيسير والتسهيل ومهارات شخصية ممتازة (غالباً ما يكون بين الحضور مجموعة متنوعة من أصحاب المصلحة ممن لم يتفاعلوا بهذه الطريقة من قبل وهناك احتمال للاختلاف والجدال؛ ويتطلب الأمر براعة ومهارة للحفاظ على استمرارية ورشة العمل وأدائها بفعالية).

■ **كاتب ورشة العمل**: هذا الشخص مسؤول عن تسجيل مخرجات الاجتماعات بطريقة رسمية. من الحكمة الطلب من أعضاء آخرين مشاركين في ورشة العمل تسجيل النقاط المهمة والاحتفاظ بنسخ احتياطية. ذلك ليس مجرد دور إداري سلبي؛ فمن الضروري أن يكون هذا الشخص قادراً على:

- فهم الطريقة والأهداف بحيث يمكنه تسجيل النقاط المناسبة بوضوح وذلك للحصول على المعلومات بالصورة المرغوب بها.
- أن يكون واثقاً بما يكفي لضبط وتيرة هذا اللقاء لضمان تسجيل النقاط المهمة وتوكيدها من قبل أصحاب المصلحة.

■ **فريق ورشة العمل**: يساعد أعضاء الفريق في الاستفسار وتركيز الاجتماع. ويجب أن يكونوا على دراية بمنهجية متطلبات الهيكلية المهمة ومعرفة بالهيكلية، وهذا أفضل دور يمكن أن يقوم به شخص جديد في هذا المجال.

■ **أصحاب المصلحة**: وهؤلاء يوفرون محتوى المخرجات.

يوضح الجدول 4-1 أصحاب المصلحة الذين يجب أن يشاركوا في ورشة العمل.

تكون مخرجات هذه الفعالية مجموعة من المتحكمات الهيكلية

مرتبة الأولويات والتي يتم نقلها إلى أهداف العمل. وغالباً ما يكون الأمر وجود القليل من المتحكمات «الحقيقية» للهيكلية نسبياً. فإذا كان لديك أكثر من خمسة إلى سبعة متحكمات متقدمة المستوى، سيكون لديك الكثير لإدارته. وفي هذه المرحلة يمكن تصفية المتطلبات لتبقي منها ما هو مهم، ولكن يجب أن تكون هذه المتطلبات قابلة للتجميع ضمن شواغل رئيسة قليلة نسبياً.

إضافة إلى الأمور الفنية المذكورة أعلاه (وربما أكثر أهمية منها)، عليك في هذا الوقت أن تجمع مجموعة من أصحاب المصلحة الذين لديهم فهمٌ مشترك لما تحاول الشركة تحقيقه من وراء بناء النظام. وغالباً ما تكون ورشة العمل هذه المرة الأولى التي يحضر فيها أصحاب المصلحة مثل هذا الاجتماع (في بعض الحالات، قد تكون تلك المرة الأولى التي يجتمعون فيها)، ومن النادر أن يكون لديهم مفهوم مشترك للهدف من المشروع بشكلٍ كامل. يجب أن يتم تسجيل هذا السياق والفهم المشترك وتركيبه في المشروع. قمنا بتفصيل ذلك في الفصل السابع.

الجدول 4-1 أصحاب المصلحة المشاركون في ورشة عمل أهم المتطلبات من ناحية الهيكلية

الوظيفة	المسؤوليات
مهندس المتطلبات	يجب أن يكون مهندسو المتطلبات على معرفة بخصائص النظام الإلزامية. هذا، وغالباً ما يكون هناك علاقة مباشرة بين الخصائص والقضايا المتعلقة بالهيكلية.
مديرو المنتج	يصور مديرو المنتج الخصائص التي يجب توفرها في المنتج لضمان تسويقه بنجاح.

راعي المشروع	يقوم راعي المشروع بتمويل عمليات برمجة وتطوير المنتج.
مصمم الهيكلية	يقوم المصممون بتعريف هيكلية النظام ووصفها وتوضيحها؛ واتخاذ القرارات الفنية وتنفيذ نموذج مطابقٍ للهيكلية. وهم مسؤولون عن إكمال مجموعة المكوّنات بنجاح وتطوير اختبارات المواصفات واختبارات القبول لمكوّنات المنتج.
مدير المشروع	يعمل مدير المشروع على إعداد خطة بدورة حياة المنتج كاملة وإدارة المتطلبات وفرق العمل لكل مشروع وتفويض فرق العمل لبرمجة وتطوير مكوّنات المنتج في مختلف أنحاء العالم.
وكلاء العميل	يوضح وكلاء العميل احتياجات مستخدمي المنتج النهائية.

4-2-2 متابعة أنشطة المتطلبات

تُعتبر ورشة العمل بحد ذاتها بمثابة الضوء الأخضر للبدء بفعاليات متطلبات الهيكلية المهمة وليست نهايتها. غالباً ما نتابع هذه الأنشطة بإجراء المزيد من التحليل التفصيلي (في مجالات يتم تحديدها خلال ورشة العمل) التي قد تتضمن إجراء المزيد من التحليلات للسوق والمزيد من تصفية السيناريوهات وتحديد الحالات الخاصة والسيناريوهات ذات العلاقة والتوثيق ومتطلبات الهيكلية المهمة الموزَّعة على نطاق واسع. من المهم أن نأخذ في الاعتبار أن هذا النهج تكراري جداً ويرتبط بأنشطة دورة حياة المشروع الباقية. ونحن نوصي بشدة بعقد ورشة العمل الأولية مبكِّراً من دورة الحياة، بيد أننا نعقد ورشات عمل متابعة قصيرة الأمد (منظم بطريقة مختلفة بعض الشيء) بناءً على فترات زمنية مُجدْوَلة ودورية لضمان أن يكون

الجميع متفقين خلال المشروع. تنخفض وتيرة هذه الاجتماعات بمرور الوقت، إذ تصبح المتطلبات والهيكلية أكثر استقراراً وتوازناً.

4-2-3 التوثيق

لا يعتبر التوثيق والإدارة أمرين بتلك الأهمية نظراً إلى أن عدد المتطلبات المهمة المرتبطة بالهيكلية قليل جداً (حتى بالنسبة إلى النظم الضخمة المعقدة) نسبة إلى المتطلبات الوظيفية. لكننا نطلب أن تكون هذه المتطلبات محددة في وثيقة عرض الهيكلية مع توضيحات لكيفية عنونتها والبدائل التي تم اعتبارها. ويكون ذلك مفيداً حين عرض وثيقة وصف الهيكلية لإحراز فهم سريع لأهم القضايا التي تم عنونتها. وينبغي أيضاً أن يكون هذا هو الحال في الأقسام الفرعية في وثيقة وصف الهيكلية. وتطفو هذه المتطلبات على السطح (بمزيد من التفصيل في بعض الأحيان) عندما يكون ذلك مناسباً. وقد تم شرح عملية توثيق الهيكلية بمزيد من التفصيل في الفصل الخامس من هذا الكتاب .

4-3 الملخص والاستنتاجات

ناقشنا في هذا الفصل منهجيتنا المتبعة لاشتقاق المتطلبات المهمة للهيكلية من أهداف المشروع. كما ناقشنا كيف يؤثِّر توزيع العمل البرمجي في عملية التصميم. ولسوء الحظ، ما زال هناك الكثير من المهارة والفن في هذه العمليات. بينما قمنا بخطوات واسعة في تنظيم وتصنيف المنهجية المنظمة لعمل ذلك، فما زال هناك اتكال كبير على الخبرة (على الخبرة غير المنتشرة) وذلك لتنفيذ البرامج بدقة. وقد وجدنا أن العملية التي عرّفناها واتَّبعناها تساعد في تركيز الاهتمام في المجالات الملائمة في الأوقات الملائمة، وذلك لتجنب بعض المزالق التي سببت فشل العديد من المشاريع السابقة،

وتؤدي هذه العملية في نهاية الأمر إلى جودة المنتج ـ والجودة العالية هي مفتاح السرعة العالية.

4-4 أسئلة للمناقشة

1. ناقش أهمية متطلبات الهيكلية المهمة بالنسبة إلى مشاريع تطوير البرمجيات الموزَّعة المراكز.
2. ما هي التقنيات المتبعة لتحديد متطلبات الهيكلية المهمة غير تلك التي ذُكرت في هذا الفصل؟

المراجع

Books

Hofmeister, Christine, Robert Nord and Dilip Soni. *Applied Software Architecture*. Boston; MA: Addison-Wesley, 2000.

Paulish, Daniel J. *Architecture-Centric Software Project Management*. Boston; MA: Addison-Wesley, 2002.

Conferences

Herbsleb, James D. and Rebecca E. Grinter. «Splitting the Organization and Integrating the Code: Conway Law Revisited." *Proceedings of the 21st Interrnational Conference on Software Engineering, (ICSE 99)*, Los Angeles, CA.

Reports

Bachmann, Felix, Len Bass and Mark Klein. *Illuminating the Fundamental Contributors* to Software Architecture Quality, Technical Report, CMU/ SEI-2002-TR-025, August 2002.

Barbacci, Mario [et al.]. *Quality Attribute Workshop* Participants Handbook, CMU/ SEI-2000-SR-001, July 2000.

Bass, Len, Paul Clements and Rick Kazman. *Software Architecture in Practice*. Second Edition. Boston: Addison-Wesley, 2003.



الفصل الخامس

الهيكلية

تقوم شركة ABC بتطوير الجيل التالي من نظامها الذي يتضمن قيوداً على الموارد وبعض متطلبات الزمن الحقيقي(*) المهمة. فقد كانت الهيكلية عبارة عن تحويل على نظام شركة ABC السابق. وقد كان من المفترض نشر النظام في بيئة عمل تصل سعة الذاكرة فيها إلى 64 ميغا بايت (وهي أكبر مما كان عليه في النظام السابق). كما يوجد متطلبات الزمن الحقيقي دقيقة وصارمة لبدء التشغيل. وقد وجدوا في شركة ABC هذه المرة أنهم يستنفذون الحد الأقصى من الذاكرة للاستعداد. وقد تغير الهيكل التنظيمي لإدارة البرمجة عبر السنوات وهم يقومون الآن بتنفيذ نماذج مختلفة من النظام في خمسة مواقع في أوروبا. وقد سبّب ذلك العديد من التفاعلات بين فرق البرمجة المختلفة جيئة وذهاباً، وهذا بدوره تطلب تنسيق كافة

(*) نظم الزمن الحقيقي هي تلك النظم التي يتم فيها الاستجابة للأوامر والتغيير مباشرة من دون إبطاء أو تأخير.

التفاعلات بين المكوّنات أثناء الاستعداد. كانت النتيجة النهائية لذلك أن البرمجة استغرقت ضعف الوقت المستحق وتجاوزت الميزانية الأصلية بنحو 80 بالمئة، ومع ذلك لم تحقق المتطلبات المستترة أو قيود الذاكرة اللازمة للاستعداد.

في المشاريع المنتظمة، تكون الهيكلية غالباً الأداة المركزية التي تربط العديد من جوانب المشروع مع بعضها بعضاً. ولا يختلف الأمر عن مشاريع تطوير البرمجيات الموزَّعة المراكز. هناك عدد من الاهتمامات الخاصة حين إعداد هيكلية البرمجيات الموزَّعة. وحين النظر في عوامل النجاح الحاسمة، سيبدو جلياً أن الهيكلية في محور العديد منها. من الواضح أن الهيكلية أمر إلزامي لفهم الطبيعة المترابطة للمهام. كما يعتمد الحد الذي يمكنك تقليل الغموض وزيادة الاستقرار إلى أقصاه على كيفية عقد وتوثيق الأنشطة المهيكلة. ذلك هو الحال أيضاً، إذ إن التصميم الأمثل للمتطلبات لا يكون هو الأمثل بالنسبة إلى الشركة التي ينبغي عليها برمجة وتطوير هذه المتطلبات. ثمة مشاكل تعترض فرق العمل النائية لتطوير مفهوم مشترك بخصوص الهيكلية، وهناك حاجة إلى التنسيق الذي تفرضه الهيكلية والذي قد يكون تحقيقه صعباً، كما يصبح تطوير الهيكلية بكفاءة أمراً صعباً بعد اكتساب المزيد من المعرفة. إضافة إلى ما تقدم، تعتمد القدرة على تنفيذ عمليات التخطيط والتقويم بكفاءة على الحصول على مدخلات معينة من أنشطة الهيكلية. ونركز في هذا الفصل على كيفية دعم عوامل النجاح الحاسمة وأخذ الشركة في الاعتبار أثناء تنفيذ أنشطة تصميم الهيكلية.

5-1 تمهيد

يُعرِّف معهد هندسة البرمجيات هيكلية البرمجية على أنها «بنية أو بُنى النظام التي تتكون من عناصر البرمجية ومكوّناتها والخصائص

الخارجية الظاهرة لهذه العناصر والعلاقات التي تربط بينها» (Bass [et al.], 2003) كما يمكن التفكير في الهيكلية على أنها مجموعة من القرارات المتخذة مسبقاً في دورة حياة التصميم التي تقيد بدورها القرارات المستقبلية. وأما في حالة تطوير البرمجيات الموزَّعة المراكز، فتُتخذ القرارات المستقبلية بصورة أكبر بواسطة فرق البرمجة الموزَّعة في عدة مواقع.

سنقوم في هذا الفصل بمناقشة كيفية تحديد القرارات المهمة التي يجب اتخاذها وكيفية فهم تأثير هذه القرارات وكيفية إبلاغ أعضاء الفريق بها.

5-1-1 احتساب متطلبات سمات الجودة

ناقشنا في الفصل الرابع من هذا الدليل تعريف «المتحكمات الهيكلية» للنظام، لكنه لم يشرح ماهيتها بصورة كافية. لقد ناقشنا عملية تعريف وتقويم متطلبات الهيكلية ذات الصلة في ما يتعلق بسياق الأعمال وذكرنا أنه يجب ألا يوجد أكثر من خمسة متحكمات هيكلية من دون ذكر التفاصيل. وقد يكون لديك من عشرين إلى خمسين متطلباً في نهاية ورشة العمل الخاصة بأهم المتطلبات من ناحية الهيكلية، حيث يتم تصنيفها حسب أهميتها بالنسبة إلى الأهداف التجارية، وقد يكون بعضها في غاية الأهمية بينما تكون أخرى غير مهمة. وعندما تحصل على التصنيف المطلوب، عليك أن تطلب من المصممين تصميم المتطلبات ذاتها بالنسبة إلى مدى صعوبة تحقيقها. ينتهي المطاف بالمتطلبات المهمة جداً للعمل التي يصعب تحقيقها لتكون «المتحكمات الهيكلية» الحقيقية. وهذا يعني أن الأمر ينتهي بها إلى ممارسة التأثير الأكبر في شكل الهيكلية. وهذا لا يعني أنك تستطيع تجاهل المتطلبات الأخرى ـ بل إنها لن تحظى بنفس التأثير في الهيكلية.

لا نقصد وصف عملية التصميم الهيكلي بتفصيل كبير هنا، بل التوجيه والإرشاد إلى الجوانب التي تعتبر حرجة بالنسبة إلى مشاريع تطوير البرمجيات الموزَّعة المراكز بمزيد من التفصيل. وهناك قدر كبير من المعرفة إلى حدٍ ما بالآليات الهيكلية للعديد من خصائص الجودة المشتركة؛ الأمر المهم هو تحليل خياراتك لتحقيق متطلبات الهيكلية المهمة (وبالتالي تحقيق أهداف العمل). إن تسجيل الأسباب المنطقية للقرارات المهمة التي تُتخذ فكرة جيدة حيث يتسنى للآخرين الانتفاع من الأفكار المضمّنة (ولا تنتهك هذه القرارات عن جهل). إن اختيار ترتيب المتحكمات والخيارات أمر متروك لك، ولكن عليك أن تأخذ في الاعتبار جميع المتحكمات والخيارات، كما ينبغي عليك إعادة النظر في القرارات السابقة، إذ إنه لا يمكن اتخاذ معظم هذه القرارات في معزل عن بعضها (نناقش المفاضلات بعد حين).

5-1-2 احتساب الهيكل التنظيمي للشركة

إذا توفر بعض الفهم للهيكلية المنوي إنجازها في الشركات النامية لدى مصممي الهيكلية، تكون لديهم فرصة لتصميم نظام قابل للتنفيذ. ويجب أن يعرف المصممون عدد فرق البرمجة التي ستعمل في المشروع والمسؤول عن إعداد هذه الفرق وميزات الأفراد، كنوع مجال المعرفة والخلفية التقنية التي يمتلكونها ومواقع وجودهم، والوحدات التنظيمية التي ينتمون إليها في الشركة وإذا ما كانوا قد عملوا معاً مسبقاً، ومدى توفرهم للعمل حالياً. من هذه المعلومات، سيتوفر لدى المصممين فكرة عن عدد وحدات العمل لديهم ونوع الجهد المطلوب لكل وحدة عمل والمهارات التي يجب احتواؤها وغير ذلك.

إذا نظرنا إلى المثال الموضح في بداية هذا الفصل، نجد أن

الحصول على تحليل للمطابقة بين الروابط والشركة يتطلب تنسيقاً معيناً. هذا، ويمكن اختيار بدائل أخرى كتخصيص ميزانيات لتوفير وحدات ذاكرة وجدولة الخوارزميات أو تقسيم النظام في ما يتعلق بالوقت والمساحة (ففي حين أن هذا يتطلب ضبط السعات التخزينية لكل قسم، فإنه يلقي الضوء بوضوح على القضايا حين نشوئها، ويوضح تأثير هذه القضايا في تلبية إجمالي المتطلبات). ثمة بديل آخر يتمثل في تغيير الهيكل التنظيمي للشركة؛ على سبيل المثال، قم بتعيين مجموعة تُعنى بالجدولة واستهلاك موارد النظام، واطلب من تلك المجموعة إدارة التنسيق بطريقة منظمة. الأمر المهم هنا هو أنه لا يمكن إدارة المخاطر بطريقة فعَّالة ما لم يتم تحديد هذه القضايا مبكراً (لمزيد من التفاصيل، انظر الفصل السادس من هذا الكتاب).

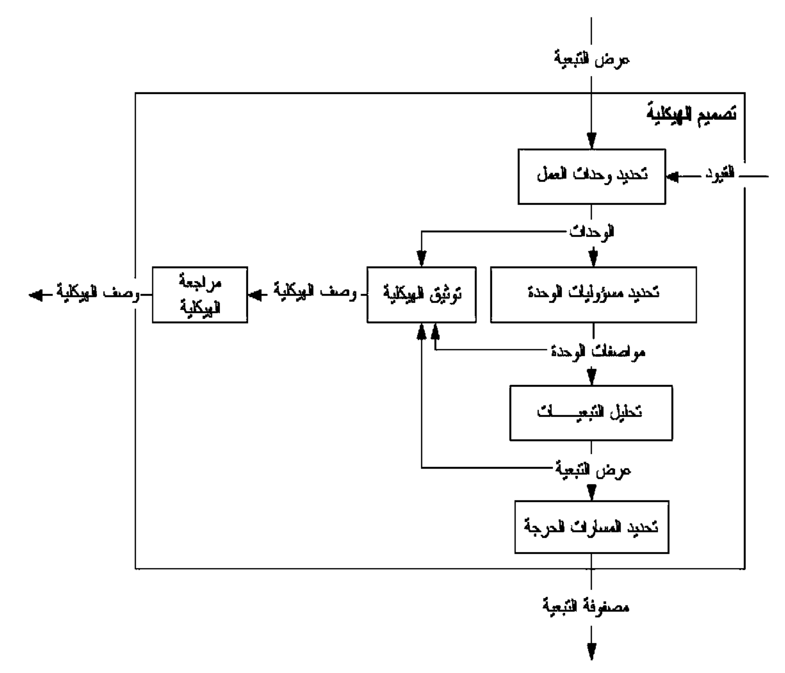
5-1-3 إجراء مفاضلات الهيكلية

كما أشرنا في القسم 5-1-1، تعتبر المفاضلات جزءاً متأصِّلاً في عملية التصميم. ولا يمكنك تحقيق الاستفادة المثلى من الهيكلية في ما يتعلق بالمتطلبات المرتبطة بها (مثلاً، إن تحقيق الأداء الأمثل يعني التأثير في قابلية التعديل)، لذا يجب إجراء حلول وسطية بحيث تشكِّل الأساس لقرارات الأعمال. ليس صحيحاً أن المصممين يكونون في مواقع اتخاذ القرارات (مع أنهم لا يتخذون هذه القرارات ضمنياً). فما لم يخبر المصممون الأشخاص المسؤولين عن وضع خطة المشروع عن أي قضية، وما لم يعرفوا تأثير هذه القضية في العمل، فإنهم لن يكونوا في موقع يسمح لهم باتخاذ القرارات المثلى. وهذه هي فائدة تطوير متطلبات الهيكلية المهمة بالطريقة التي أوصينا بها في الفصل الرابع. وهي وسيلة للمصممين والأشخاص المعنيين بالأعمال للتواصل واتخاذ القرارات التقنية التي تؤثر في خطوط الإنتاج في العمل. لذا، من المهم إعطاء أصحاب المصلحة

الملائمين فرصة فهم القضايا والخيارات لتوفير مدخلات للمشروع. ونوصي بعقد اجتماعات دورية مجدولة لتحقيق ذلك.

5-2 تصميم النظام

يوضح الشكل 5-1 مخطط سير تصميم الهيكلية. كما ذُكر سابقاً، لا ننوي الدخول في التفاصيل لوصف كيفية تصميم النظام. وبدلاً من ذلك، نركز على جوانب عملية التصميم المرتبطة بتطوير البرمجيات الموزَّعة المراكز أو توفير مدخلات للعمليات الأخرى كعملية التخطيط. وبقراءة الفصول التالية، ضع في اعتبارك أن تلك ليست قائمة متسلسلة؛ ففي حين أننا قد وصفنا هذه الأنشطة بصورة مستقلة من أجل الوضوح، غير أنها مترابطة عملياً وتحدث بشكل متكرر.



الشكل 5-1: مخطط عمل لتصميم الهيكلية

5-2-1 تحديد وحدات العمل

في بعض المراحل، يجب تقسيم النظام إلى وحدات وظيفية يمكن تخصيصها إلى فرق العمل لبرمجتها وتطويرها. ولن نخوض في تفاصيل كيفية عمل ذلك (تعتبر هذه العملية في الكثير من الأحيان «فناً»، فضلاً عن كونها نشاطاً من أنشطة هندسة البرمجيات)، لكننا سنركز على النقاط المهمة المرتبطة بتطوير البرمجيات الموزَّعة المراكز أو التي لها تأثير في فعاليات أخرى موضحة في مكان آخر في هذا الكتاب.

في كتب التصميم، تكون المصطلحات الخاصة التي يستخدمها المؤلفون مربكة؛ ففي بعض الحالات، يبدو الأمر كأن المؤلفين أنفسهم غير متأكدين من معاني هذه المصطلحات (في واقع الأمر، وجدنا كوننا خمسة مؤلفين لهذا الكتاب بأننا غالباً ما استخدمنا تعبيرات غير متجانسة)؛ وهذا الأمر وارد جداً في أبحاث ودراسات هيكلية البرمجيات. وسنحاول وصف مقصدنا من الاستخدام المفرط للمصطلحات. عندما نتحدث في هذا القسم عن تقسيم النظام إلى وحدات وظيفية، سنتكلم عن الوحدات الثابتة (نستخدم التعبير «وحدة» لتفيد ضمنياً معنى وحدة شيفرة البرمجة الثابتة). في حالات كثيرة، سيكون هناك مطابقة معقولة بين عناصر وقت التشغيل والوحدات الثابتة أو الحزم (وقد يكون هذا هو سبب الإرباك)، لكنها لن تكون مطابقة مثالية. سيكون هناك وحدات ثابتة أكثر من وحدات وقت التشغيل، إذ قد لا تتوفر بعض الكوائن والوحدات كطبقات الأدوات البرمجية (utility classes) كتوفر كوائن وقت التشغيل.

هناك عدة أمور تحكم كيفية تحديد التقسيم الوظيفي للنظام، على سبيل المثال، الخبرة السابقة والمتطلبات المهمة للهيكلية، والموارد المتوفرة والأنماط ومعلومات الشركة. نظرياً، ستأخذ بعين

الاعتبار متطلبات الهيكلية المهمة ذات الصلة. أما عملياً، فتعتبر الخصائص التالية من الأمور التي تؤثر في عملية التحليل: قابلية التعديل وقابلية التوسيع وقابلية التغيّر، إضافة إلى مدى التوفر. إذا كان الأمر أنك تقوم ببرمجة خط الإنتاج، وأنك قمت بتعريف أوجه الشبه والمتغيرات، فإنك سترغب في التأكد من تضمين الأقسام المتغيرة في نظامك وذلك لإتاحة الفرصة لإنشاء المنتج المرغوب به بسهولة. إضافة إلى ما تقدم، إذا كان هناك احتياجات مختلفة لتوفر عدد متنوع من مزايا النظام فقد ترغب بعزل هذه المزايا (وهذا ينطبق أيضاً على أداء الزمن الحقيقي). ستحكم أنواع متطلبات الهيكلية المهمة عملية تقسيم النظام، لذا سترغب في التأكد من أن نظامك قد جُزِّئ إلى وحدات تنفيذية. ويتضمن الفصل الثامن بعض الإرشادات والمبادئ التوجيهية عن تحديد السعة المثلى للوحدات، ولكن تقسيم النظام يحتاج بشكل أساسي إلى تسهيل عملية التقسيم.

عندما نقوم بتعريف الوحدات القابلة للتنفيذ، يجب أن نأخذ في الاعتبار فرق العمل التي ستعمل على برمجة النظام. يجب ألا ينسى مصممو الهيكلية الفريق الذي سيقوم بعملية التقسيم وذلك لضمان وجود فريق عمل مناسب للوحدات التي يتم تحديدها (يتم عمل ذلك أثناء عملية المراجعة). إذا لم يتوفر فريق مناسب، يجب أن يحاول المصممون إيجاد بدائل. فإذا كانت البدائل أقل من الفريق المثالي الذي يطمحون له في ما يتعلق بمتطلبات الهيكلية المهمة الأخرى، عندئذ قد يتطلب الأمر إجراء مفاضلة بين البدائل التي قد تكون على هيئة متطلبات مهمة للهيكلية غير رسمية أو بإعادة هيكلة بعض جوانب الشركة (سنناقش عملية إعادة هيكلة البدائل في الفصل السادس ـ المخاطر)، أو قد تتم المفاضلة بزيادة الميزانية أو جدولة المشروع. نتيجة لتنفيذ هذه الفعالية سيتكون لديك تحديد للوحدات التي تُشكِّل النظام على الأقل، وبعض أنواع تقويم سعات الوحدات

ووصفاً لمسؤوليات كل وحدة من الوحدات. إذا تطلب الأمر اتخاذ قرارات حاسمة سيكون من الجيد توثيق هذه القرارات وما تحاول إنجازه (مرجع لمتطلب أو اثنين من المتطلبات) والبدائل التي تم بحثها والأساس المنطقي الذي سيُعتمد لاختيار البدائل.

كلما تقدم المشروع، فسيتطور هذا العمل بتوفر تعريفات كاملة وثابتة للتقسيم الوظيفي ومخططات النشاط التي تحدد كيفية تحقق المزايا والخصائص في الوحدات ومواصفات واجهة التطبيق والعلاقات المترابطة. سنناقش عملية توثيق هذه الأهداف وغيرها في القسم التالي.

5-2-1-1 المشاركون

يعرض الجدول 5-1 قائمة المشاركين الذي يقومون بتحديد وحدات العمل ومسؤولياتهم الأساسية.

الجدول 5-1: المشاركون في نشاط تحديد وحدات العمل

الوظيفة	المسؤوليات
مصمم الهيكلية	المصمم مسؤول عن هيكلية النظام، وبالتالي فهو مسؤول عن التقسيم الوظيفي.
مدير المشروع	يوفر مدير المشروع معلومات للسعات التخزينية المثالية للوحدات بناءً على أنشطة التخطيط. كما يتوجب على مدير المشروع تقديرات السعات التخزينية لكل وحدة. كما يوفر المعلومات عن مواصفات فرق العمل. وهذا أمر مفيد لضمان توفر فرق عمل ذات مهارات وخبرات كافية لبرمجة وبرمجة الوحدات التي يعرفها مصممو الهيكلية.

مهندس المتطلبات	يساعد مهندس المتطلبات في توضيح متطلبات العمل والمشاركين في مراجعة التصميم أو الأنشطة الأخرى لضمان أن المتطلبات معرّفة بشكل كافٍ.

5-2-2 تحديد مسؤوليات الوحدة

سيحتاج مصممو الهيكلية في وقت من الأوقات إلى تحديد أجزاء النظام التي تعمل مع بعضها بعضاً، وذلك لتحقيق خصائص النظام المحددة في المتطلبات (انظر الفصل الثالث من هذا الكتاب). وهناك ثمة اقتران ضعيف بين هذا النشاط وتحديد وحدات العمل (التي تم وصفها في القسم السابق). قبل تنفيذ هذا النشاط، يجب توفر نموذج للمتطلبات وتعريف أولي لوحدات النظام. وهذا النشاط تكراري (كما هو الحال في معظم الأنشطة الأخرى) ما يعني أنه سيتم تنقيح نموذج المتطلبات والوحدة الثابتة في النظام بالتوازي مع هذا النظام. من الواضح أن المتطلبات المحددة التي يتم العمل فيها يجب أن تكون ثابتة بشكل منطقي قبل تنفيذ هذا النشاط.

ستتطوَّر هذه الوحدة الثابتة كنتيجة لتنفيذ هذا النشاط. وبالنظر إلى المسؤوليات بمزيد من التفصيل، ستثار تساؤلات حول إذا ما كانت المسؤوليات ستستمر، وأن إجراء تعديلات على الأهداف الثابتة أمر لا مفر منه.

ناقشنا في الفصل الثالث كيفية تنقيح الخصائص والمزايا التي حددتها مجموعة التسويق في المتطلبات الواقعية. ويمكننا تحديد المتطلبات على مستوى النظام حالما تتوفر المتطلبات الواقعية التفصيلية (فكر بهذه المتطلبات على أنها الواجهات الخارجية للنظام). ثم نحتاج إلى إسناد المسؤوليات لمختلف الوحدات في النظام بحيث

تتحقق النتائج المرجُوَّة. لتعزيز رغبتنا في الإبقاء على عملية التتبع، قمنا باستخدام لغة النمذجة الموحدة (نقوم بإنشاء قسم للتصميم لنفس نموذج المتطلبات). كما نستخدم مخططات التفاعل (وهي مخططات التسلسل(*)) لتوثيق المتابعات خلال النظام. نقوم بتحديد واجهات الاستخدام والمناهج التي يجب فهمها لتنسيق الوحدات في النظام. ستجد هذه المناهج طريقها في آخر الأمر إلى الجدول الزمني (الموضح في الفصل السابع ـ التخطيط). يوضح الشكل 5-2 مثالاً لمخططات التسلسل.

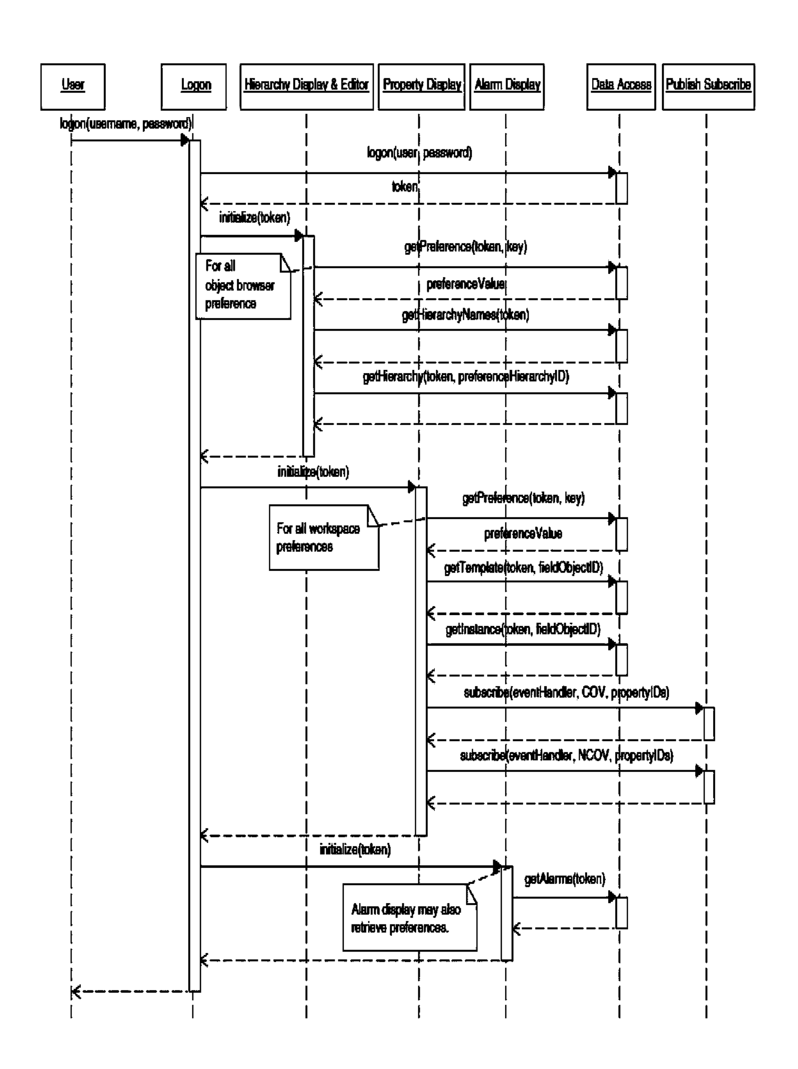
مخرجات هذا النشاط هي عبارة عن مجموعة من مخططات التسلسل التي تصف كيفية تحقق المتطلبات في النظام. هذا، وسيتطور نموذج التصميم كجزء من هذه العملية، وسيكون هناك العديد من المناهج التي تم تحديدها بالإضافة إلى أنواع البيانات النظرية المرتبطة التي يجب الاتفاق عليها. سيتطور الهدف الثابت أيضاً، كما ستنشأ القوانين التي تحدد مسؤوليات الوحدة. إذا كانت هذه القوانين مهمة (بمعنى وجود تأثير أو أكثر في أهم المتطلبات من ناحية الهيكلية)، يجب أن تكون واضحة صريحة وموثَّقة.

5-2-2-1 المشاركون

يقوم فريق الهيكلية بتنفيذ هذا النشاط. وغالباً ما يقوم أعضاء

(*) المخطط التسلسلي في لغة النمذجة الموحدة هو نوع من المخططات التفاعلية التي توضح كيفية تفاعل العمليات التي تعمل مع بعضها بعضاً وبأي ترتيب. ويعرض هذا المخطط العمليات المختلفة على شكل خطوط عمودية متوازية والرسائل المتبادلة بين العمليات على شكل أسهم أفقية وفق ترتيب حدوثها. وهذا يوفر عرضاً لخصائص سيناريوهات الزمن الحقيقي بشكل رسم بياني.

فريق الهيكلية ببعض المشاركة أو المراجعة لضمان تحقيق المتطلبات كما هو متوقع.

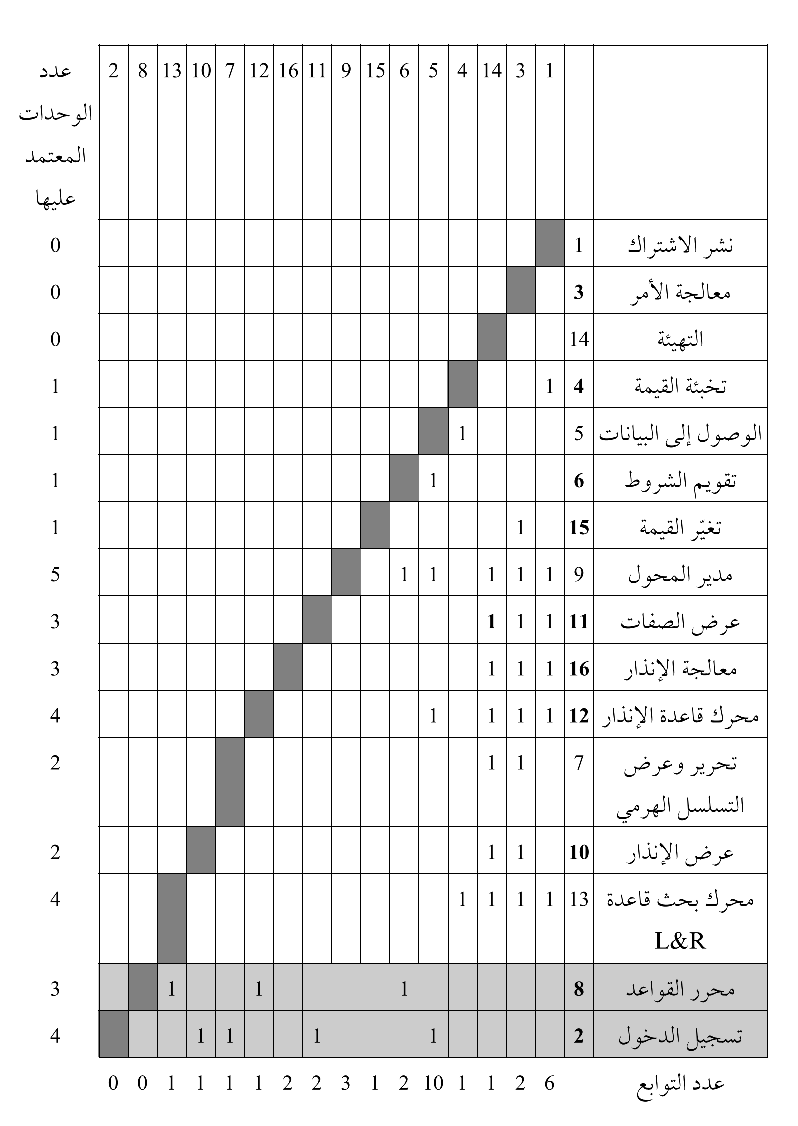


الشكل 5-2: مثال على مخطط توزيع متطلبات النظام

5-2-3 تحليل الروابط

كما ذُكر سابقاً في هذا الفصل وفي فصول أخرى، من الصعب تسهيل التنسيق بين الفرق التي تعمل في عدة مواقع جغرافية. وتملي النقاشات التي نجريها أثناء تصميم النظام مقدار التنسيق اللازم خلال المشروع. من الواضح أن بعض الأمور كالتواصل الصريح والضمني بين الوحدات تنطوي على روابط، ولكن يمكن أن يكون لديك أنواع أخرى من الروابط الأقل وضوحاً.

نعرّف الترابط في إطار هذا السياق كمظهر من مظاهر الوحدة التي تعتمد على وحدة أخرى أو على مظهر من مظاهر النظام. وقد يكون ذلك ترابطاً تركيبياً كالاعتماد على طريقة محددة أو على تنظيم محدد للبيانات أو سلوك دلالي أو سلوك زمني أو سلوك خاص بالموارد. في مرحلة من المراحل، ستعتمد كل وحدة على غيرها من الوحدات تقريباً (مثلاً إذا تشاركت جميع الوحدات في وحدة المعالجة المركزية فسيكون لها روابط). الحل هو بتعريف الروابط التي ستتطلب تنسيقاً بين فرق العمل. وهذا أمر ضروري لتحديد جدوى قيام شركتك ببرمجة الحلول التي يتم تصميمها (سيناقش ذلك في الفصل السادس ـ المخاطر). ومن الطرق المفيدة والعملية في هذا السياق النظر إلى العلاقات بين التواصل الضمني والصريح (مثلاً نشر الاشتراك) والعلاقات بين الاستخدامات (مثلاً تستخدم إحدى الوحدات الطبقات البرمجية من وحدة أخرى) وتطوير مصفوفة الترابط (انظر: الشكل 5-3). وكخطوة ثانية، انظر إلى متطلبات زمن التشغيل الحرجة من قائمة متطلبات الهيكلية المهمة (مثلاً متطلبات الأداء أو التوفر)، ثم قم بتحديد أجزاء النظام التي تشترك في تحقيق هذه المتطلبات. وغالباً لا تتم هذه الطريقة في النظر إلى النظام لكنها توفِّر فهماً عميقاً يساعد في تجنُّب التكاليف الساذجة التي قد تتكلفها الشركة.



		1	3	14	4	5	6	15	9	11	16	12	7	10	13	8	2	عدد الوحدات المعتمد عليها
نشر الاشتراك	1																	0
معالجة الأمر	**3**																	0
التهيئة	14																	0
تخبئة القيمة	**4**	1																1
الوصول إلى البيانات	5				1													1
تقويم الشروط	**6**					1												1
تغيّر القيمة	**15**		1															1
مدير المحول	9	1	1	1		1	1											5
عرض الصفات	**11**	1	1	**1**														3
معالجة الإنذار	**16**	1	1	1														3
محرك قاعدة الإنذار	**12**	1	1	1		1												4
تحرير وعرض التسلسل الهرمي	7		1	1														2
عرض الإنذار	**10**		1	1														2
محرك بحث قاعدة L&R	13	1	1	1	1													4
محرر القواعد	**8**						1					1			1			3
تسجيل الدخول	**2**					1				1			1	1				4
عدد التوابع		6	2	1	1	10	2	1	3	2	2	1	1	1	1	0	0	

القائمة التفسيرية للشكل:

1 بالخط الغامق = روابط الاستخدام؛ 1 بالخط العادي = روابط طبيعية

الشكل 5-3: مثال على مصفوفة الروابط

نحصل من هذا النشاط على ما يسمّى عرضاً لروابط النظام. ويجري تنفيذ معظم هذا العمل أثناء مرحلة التطوير (elaboration phase) لدعم التوجه نحو توزيع العمل على الفرق. إذا كان ذلك أكثر خطورة من منظور تطوير البرمجيات الموزَّعة المراكز (انظر الفصل السادس من هذا الكتاب)، يجب إبداء المزيد من الاهتمام بالروابط في النظام بتطور التصميم. إذا كانت الروابط ناشئة في المناطق الحرجة من النظام التي تنطوي على درجة عالية من التنسيق، فإنه يجب تحديد ذلك مبكراً ما أمكن حتى تتخذ الخطوات المناسبة (انظر الفصل السادس من هذا الكتاب). في هذه الحالات، يمكن اختيار آليات هيكلية بديلة.

5-2-3-1 المشاركون

يعرض الجدول 5-2 المشاركين الذين ينفذون عملية تحليل الروابط بالإضافة إلى مسؤوليات كل منهم.

الجدول 5-2: المشاركون في نشاط تحليل الروابط

الوظيفة	المسؤوليات
مصمم الهيكلية	مسؤول عن تحديد الروابط
مهندس المتطلبات	يجب أن يكون موجوداً لتوضيح المتطلبات

5-2-4 تحديد المسارات الحرجة

يجب تحديد بعض الاحتياجات البرمجية في بعض المراحل وذلك لوضع خطة برمجة وحدات العمل. ونقوم بذلك من خلال

تحديد «المسارات الحرجة»(*) في البرمجية. بمعنى آخر، تشير أي وحدة تتضمن روابط استخدام إلى أن المستخدم في تلك العلاقة يمتلك معرفة أساسية في «تمثيل الاستخدام». ولأن فرق البرمجة المتباعدة مسؤولة عن التصميم التفصيلي لمظاهر الوحدات الداخلية، يجب أن يتم تصميم (وحتى برمجة) وحدات «تمثيل الاستخدام» قبل تصميم وحدات «المستخدمين». ويمكن تحديد المسارات الحرجة بالنظر إلى النظام بهذه الطريقة.

يتم تنفيذ هذا النشاط للتحضير لعملية التخطيط، ولذلك تبدأ متأخرة في مرحلة التطوير. يجب أن تكون المسؤوليات ثابتة (بالنسبة إلى الجزء الذي تتم برمجته على الأقل)، كما يجب الانتهاء من التخطيط المتقدم. يجب توضيح الخصائص التي سيتم تخصيصها في إصدارات مفردة والروابط بين هذه الخصائص. ينتج من هذا النشاط نوع من مصفوفات الروابط التي توضح الروابط البرمجية بين وحدات النظام.

5-2-4-1 المشاركون

ينفذ هذا النشاط أعضاء الفريق المختص بالهيكلية.

فكرة مفيدة:

أبقِ حجم فريق الهيكلية صغيراً ما أمكن. وعلى الرغم من أن عدد الأدوار والمسؤوليات محدد، غير أنك يجب

(*) (CPM = critical path method). طريقة المسار الحرج (CPM) أو تحليل المسار الحرج هي خوارزمية رياضية لجدولة مجموعة أنشطة المشروع. وهي أداة مهمة لإدارة المشاريع بفعالية. تساعد هذه الأداة في توضيح ترتيب تنفيذ الأنشطة والأنشطة التي يمكن تنفيذها في نفس الوقت وعدد الأفراد الذين يلزم توظيفهم لتنفيذ أنشطة المشروع.

أن تحد من حجم فرق الهيكلية، إذ تعمل الفرق الأصغر حجماً بصورة أسرع، وتساعد في الحد من الاستثمار المطبق في مرحلة التصميم المتقدم. يسبب تعيين عدد كبير من المصممين في فريق المشروع ما يمكن أن نطلق عليه وضع «كثرة الطباخين» الذي قد يؤدي إلى «شلل التحليل». أكثر الأدوار أهمية هي الأدوار التي يضطلع بها مدير المشروع والمصمم، إذ يلعبان دوراً مهماً في تقدم المشروع بكفاءة أثناء مرحلة التطوير. غالباً ما يعمل الأشخاص الذين يلعبون أدواراً رئيسية في المشروع كمراقبي عمليات لأنهم الأكثر خبرة ولأنهم عملوا مسبقاً في مشاريع مشابهة.

5-2-5 توثيق الهيكلية

سينتج من تنفيذ بعض الأنشطة المذكورة أعلاه عروض خاصة للنظام قيد الإنتاج. ونذكر لاحقاً بعض العروض الإضافية للنظام ونناقش صلتها الوثيقة من وجهة نظر مشاريع تطوير البرمجيات الموزَّعة المراكز. لن نناقش قضايا عامة ولا أهداف عملية توثيق هيكلية النظام، بل نوصي بالتوثيق كتابةً. في كثير من الأحيان، تُوثّق الهيكلية كتابةً من دون التفكير في القارئ، فغالباً ما نرى وثيقة ضخمة خاصة بالهيكلية تعطى لأي شخص يرغب في معلومات عن الهيكلية. ولا يكون القصد من وراء هذه الوثيقة القراءة من الغلاف إلى الغلاف كالروايات؛ بل يبحث القرَّاء عن مواضيع معينة يقرأونها، لذا يجب أن يركِّز المؤلفون على احتياجات القراء قبل الشروع في الكتابة وعرض بعض الإرشاد والتوجيه حول كيفية الوصول إلى المعلومات الملائمة. ولمزيد من المعلومات عن توثيق هيكلية البرمجيات، يرجى الاطلاع على (Clements [et al.], 2002).

5-2-5-1 عروض التنفيذ

تُصور عروض تنفيذ النظام كوائن زمن التشغيل والآليات التي تستعملها هذه الكوائن أثناء تفاعلها مع بعضها بعضاً. وهي تصور مكوّنات النظام كالعمليات والكوائن ومستودعات تخزين البيانات والروابط التي تمثل آليات التفاعل بين المكوّنات. وهذه المكوّنات والروابط هي العناصر الممثلة في نوع العرض.

يفيد هذا النوع من العروض كلاًّ من:

- المختصين بمجال العمل والهيكلية والانضباط والتكنولوجيا الذين يستطيعون بيان صفات الهيكلية ومتطلبات خصائص الجودة التي يجب أن يلتزم بها النظام.
- أصحاب المصلحة (من خارج الشركة) كالعملاء ومقيِّمي المشروع الذين يستطيعون فهم عناصر التنفيذ الرئيسة للنظام (بما في ذلك مصادر البيانات المشتركة المهمة) وتفاعلاتها ـ وهي بالتالي تعمل كوسيلة للتحقق والمصادقة على خصائص المشروع.
- مسؤولي صيانة المشروع الذين يستطيعون الحصول على نظرة عامة عن النظام كنقطة بداية لتعديله وتوسيعه مستقبلاً.

لهذا السبب، يُستخدم عرض التنفيذ لتحديد الروابط المهمة بين المكوّنات الموصوفة في وقت التشغيل. ويوفِّر التتبع بين عرض تخصيص العمل الثابت وعرض التنفيذ بصيرةً إضافية نحو روابط زمن التنفيذ التي نجدها بين وحدات العمل المخصص لفرق البرمجة العديدة.

5-2-5-2 عروض التطبيق

توضح عروض التطبيق أجزاء النظام الفريدة والمجموعات غير

المتداخلة من الوحدات التنفيذية القابلة للتقسيم هيكلياً. ومن الأمثلة على وحدات التطبيق: (namespaces) في لغة البرمجة «NET». والحزم في لغة جافا Java. ثمة أهداف عديدة لهذه العروض في النظام، منها:

- تبيِّن كيفية تحليل شيفرة البرمجة المصدرية.
- تبيِّن العلاقات المختلفة بين الوحدات التنفيذية.
- اعتبار الوحدات التنفيذية التي تحقق وظائف النظام كما تم تقسيمها في عروض التنفيذ، وبالتالي تحقيق متطلبات مواصفات الجودة.
- ضمان توفر كافة المدخلات الضرورية في مرحلة وضع خطة البرمجة لوضع خطة الجدوى.

يفيد هذا النوع من العروض كل من:

- مدير المشروع الذي يجب أن يجدد مهام العمل التي تم فصلها عن بعضها بعضاً بمنطقية تسهل تفويضها عبر فرق البرمجة العديدة الموزَّعة والمنفصلة عن بعضها بعضاً.
- المختصين بمجال العمل وبالهيكلية الذين يجب أن يكونوا قادرين على بيان أكثر خصائص الهيكلية اعتماداً على عرض النظام (مثلاً خاصية قابلية التعديل) وتبرير ذلك.
- مدير المشروع والمختصين بمجال العمل والهيكلية والانضباط الذين يستخدمون هذه العروض للتحقق من حجم ودرجة تعقُّد مهام العمل بحيث يكونون قادرين على تحديد مستوى التحليل الملائم وتخصيصه لفرق عمل معينة.
- المختص بالتواصل الذي يعرف الطريقة الأفضل لتنظيم وثائق التعاون الفنية كمشروع Wiki (انظر الفصل الثاني عشر من هذا الكتاب).

- مدير التهيئة المسؤول عن صيانة الإصدارات القديمة والإصدار الحالي من الوحدات المنظمة في مجموعات وظيفية ومتناسقة يمكن رزمها، وهو قادر على إنتاج إصدار حالي من النظام.
- فاحصي النظام الذين يستخدمون الوحدات لإنشاء حالات الاختبار وتنفيذ الاختبار اللازم.

5-2-5-3 المقارنة بين طرق العرض المختلفة

تستخدم فرق البرمجة الموزَّعة في عدة مواقع طرقاً عديدة لعرض الهيكلية؛ ولتسهيل ذلك، لا بُدَّ من توفُّر نموذج منطقي مشترك للهيكلية لدى هذه الفرق. ومن المهم ضمان دقة وسهولة التصفح والتنقل بين هذه العروض في جميع مراحل المشروع. وهذا يعني ضرورة تتبع عمليات تنفيذ العروض وعروض الوحدة ووثائق التصميم الفنية التي أنشئت لاحقاً في مرحلة التطوير. وهذا أمر ضروري لسببين:

- تتيح المجال لمدير المشروع والمختص بالهيكلية والمختص في مجال العمل الالتقاء معاً وفهم الروابط، ليس فقط بين مكوّنات العروض وحسب بل خلال العروض نفسها، وهذا يصبح أداةً مهمَّةً أثناء تخصيص التنفيذ أو وحدات التنفيذ في مواقع البرمجة النائية بهدف تقليل الروابط خلال هذه المواقع.
- تستطيع فرق البرمجة التي تعمل عن بُعد النظر إلى العرض الذي يعتبر أساس خطة البرمجة وتتبع تخصيص وحدات البرمجية من خلال عدة عروض لفهم مهام العمل المخصصة فهماً أفضل ـ وهذه خطوة حاسمة في تطوير فهم شامل ودقيق للعمل الذي يجب إنجازه في كل فترة برمجية تكرارية.

5-2-6 مراجعة الهيكلية

نوصي بإجراء مراجعات رسمية ـ واحدة على الأقل ـ قبل نهاية مرحلة التطوير. نستخدم أسلوباً يعتمد على طريقة تحليل مبادلات الهيكلية المتبع في معهد هندسة البرمجيات وهذا الأسلوب يعرف بـ (ATAM) (Clements [et al.], 2001). ففي حين أننا لن نعيد خطوات أسلوب (ATAM) هنا، لكن نرى أنه من الضروري أن نقوم بالمراجعة بنفس الطريقة كما في ورشة عمل متطلبات الهيكلية المهمة. بمعنى آخر، قد ترغب في مراجعة الهيكلية في ما يخص أهداف المشروع وقدرة الشركة على تطوير الهيكلية. لعمل ذلك، عليك أن تُشرِك أصحاب المصلحة في ذلك على نطاق واسع. يتم دعم وتعزيز هذه العملية عن طريق تنفيذ أعمال ورشة عمل لأهم المتطلبات من ناحية الهيكلية. وهذا يساعد أصحاب المصلحة في فهم ما يتوقع إنجازه والأهداف التي أعلنت عنها الشركة ومتطلبات الهيكلية المهمة التي تم تحديدها والغرض من الاجتماعات التي يتم عقدها. وتختلف مخرجات عملية المراجعة، إذ قد تكون عبارة عن المخاطر والأمور غير الخطرة والمبادلات والنقاط الحساسة في الهيكلية. لمزيد من المعلومات حول هذه العملية، الرجاء مطالعة (Clements [et al.], 2001).

5-3 الملخص والاستنتاجات

عُني هذا الفصل بأمور هيكلية النظام المرتبطة بمشاريع تطوير البرمجيات الموزَّعة المراكز. من المهم تحسين الهيكلية بحيث يسهل توزيعها على فرق العمل التي تعمل في عدة مناطق جغرافية. يجب أن يقوم مصممو الهيكلية بإنشاء وحدات العمل مع

الأخذ في الاعتبار إمكانيات أعضاء فرق العمل وقدراتهم على العمل مع بعضهم بعضاً ومستوى التنسيق المطلوب بينهم. ستساعد عملية تحديد الوحدات ومسؤولياتها والروابط بينها مديري المشروع في إنشاء وحدات العمل. كما تساعد عملية تحليل الروابط في تحديد المسارات الحرجة، وبالتالي جدولة عملية برمجة هذه الوحدات. وبما إن الهيكلية هي أمر حرج لنجاح المشروع ـ خصوصاً إذا كان العمل موزعاً في عدة مناطق جغرافية ـ يتوجب مراجعتها مع أصحاب المصلحة. كما يجب توثيقها لمساعدة فرق العمل ذوي العلاقة على فهم الهيكلية ومشاركة الرؤى ذاتها. وكما أكدنا في الفصل الرابع من هذا الكتاب، فإن الهيكلية ذات الجودة العالية تقطع شوطاً طويلاً نحو إنتاج منتج عالي الجودة. تساعد التقنيات الموضحة في هذا الفصل في التركيز على الامتياز التقني في مراحل مبكرة من دورة حياة تطوير البرمجية.

5-4 أسئلة للمناقشة

1. ما هو المغزى من الهيكلية والهيكل التنظيمي للشركة في مشاريع تطوير البرمجيات الموزَّعة المراكز؟
2. ما هي العملية التي يستطيع مدير المشروع إجراؤها ليتمكن من إنشاء وحدات العمل وجدول الجدوى لتطوير البرمجيات الموزَّعة المراكز لمنتج برمجي؟
3. صف بعض طرق العرض المستخدمة لتوثيق الهيكلية بالإضافة إلى تلك المعرفة في هذا الفصل. ما هو مغزى هذه الطرق، إن وجدت، بالنسبة إلى مشاريع تطوير البرمجيات الموزَّعة المراكز؟

المراجع

Books

Bass, Len, Paul Clements and Rick Kazman. *Software Architecture in Practice*. Second Edition. Boston, MA: Addison-Wesley, 2003.

Clements, Paul [et al.]. *Documenting Software Architectures: Views and Beyond*. Boston, MA: Addison-Wesley, 2002.

———————, Rick Kazman and Mark Klein. *Evaluating Software Architectures: Methods and Case Studies*. Boston, MA: Addison-Wesley, 2001.



الفصـــل الســادس

تحليـــل المخاطــر

عندما كان جو يعمل في مشروع (BAS)، شارك في دورة تدريبية حول إدارة المخاطر. نصح المُحاضر فريق إدارة المشروع بالتركيز على بعض المخاطر التي قد يكون احتمال حدوثها كبيراً وقد تؤثر بصورة سلبية في المشروع. فكّر جو بكافة المخاطر المحتملة المرتبطة بتطوير البرمجيات الموزَّعة المراكز على مستوى كبير لكن شركته تفتقر إلى الخبرة الكافية في مجال الاستعانة بمصادر خارجية، أدرك أن هناك الكثير من المخاطر المقلقة التي قد تؤثِّر في المشروع.

بالنظر إلى درجة تعقيد مشروع (BAS)، كان من الواضح وجود العديد من المشاكل المحتملة. في واقع الأمر، كانت المشاكل كبيرة بحيث تهدد نجاح المشروع. هل يوجد وقت معين تعالج فيه هذه القضايا؟ إذا كانت الإجابة بالإيجاب، هل يمكن اتباع استراتيجيات للحد من احتمالات تحقق هذه المخاطر؟

نناقش في هذا الفصل إدارة المخاطر المتعلقة بتطوير البرمجيات الموزَّعة المراكز. تتشابه عملية إدارة المخاطر في مشاريع تطوير البرمجيات الموزَّعة المراكز مع إدارة المخاطر في مشاريع البرمجيات المجمَّعة التقليدية في العديد من النواحي، غير أن هناك بعض الاعتبارات الخاصة. نسلط الضوء هنا على طرق اختلاف إدارة المخاطر في مشاريع البرمجيات الموزَّعة، حيث نفترض أن القارئ لديه معرفة عامة عن إدارة المخاطر.

إن إدارة المخاطر هي المسؤولية الأساسية لمدير المنتج (أو البرنامج). وقد يساعده في ذلك موظفو ضمان الجودة أو خبراء العمليات، لكن جميع أفراد فريق العمل مسؤولون عن تحديد المخاطر المحتملة والإجراءات اللازمة للحد منها. يجب وضع خطة لإدارة المخاطر أثناء مرحلة التطوير، لكن تنفيذها ومراقبة تطور العمل بها تتم بعد توثيق أول مخاطرة.

6-1 تمهيد

نقدم في هذا القسم تمهيداً عن إدارة المخاطر، ونناقش مصادر المخاطر المرتبطة بمشاريع تطوير البرمجيات الموزَّعة المراكز خصوصاً. هذه المعلومات غير مترابطة في بعض المعاني في هذا الكتاب، إذ نناقش في كل قسم قضايا محددة (أي المخاطر) ترتبط بالأنشطة قيد النقاش. كما نناقش طرق تحديد وعنونة هذه القضايا (تخفيف حدة المخاطر).

لماذا خصصنا إذاً فصلاً خاصاً لإدارة المخاطر؟ ألم نتناول هذه القضايا بشكل كافٍ في هذا الكتاب؟ تعتبر إدارة المخاطر نشاطاً حرجاً في إدارة المشاريع بذاتها، حيث تعتبر أهداف العمل المحفِّز لاتخاذ القرارات المتعلقة بإدارة المخاطر. غالباً لا يقوم المديرون

بعمل شيء سوى إدارة المخاطر بطرق عديدة. وفي هذه الحالة، من المنطقي وجود منهجية منتظمة لمراقبة مدى جودة إدارة المخاطر وأين يجب أن يتركز الاهتمام الإضافي. ثمة أمور محددة في مشاريع تطوير البرمجيات الموزَّعة المراكز لا تكون واضحة إلى أن يظهر تأثيرها في المشروع. لقد شهدنا من قبل العديد من المشاريع التي فشلت لأنها لم تلقَ الاهتمام الكافي مبكراً بما يكفي لتفعيل عوامل معينة في مشاريع تطوير البرمجيات الموزَّعة المراكز. لقد ذكرنا هذه العوامل في هذا القسم.

6-1-1 ما هي المخاطر؟

هناك تعريفات عديدة للمخاطر كمعظم المصطلحات الفنية. أساسياً، المخاطرة هي احتمالية المعاناة من الخسارة (Dorofee [et al.], 1996). الأمر المهم هنا هو إدراك أن هناك حالة من «عدم التأكد» من أن الخسارة ستسبب نوعاً من المعاناة. إذا حدثت الخسارة فعلاً أو في حالة التأكد من حدوثها، لا يمكن القول إنك تتعامل مع مخاطرة ـ بل إنك تتعامل مع مشكلة. بالنظر إلى النقاش المفتوح حول مشروع (BAS)، لا نقوم بوصف المخاطر بل نصف المشاكل التي حدثت. المشاكل تكون مخاطر في المراحل المبكرة من المشروع وقبل التأكد من حدوث هذه المشاكل (أي عندما كانت مجرد احتمالات). في مشاريع البرمجيات، تكون الخسائر التي قد تسبب معاناة على شكل زلات في الجدول الزمني للمشروع أو تكاليف زائدة أو جودة المنتج النهائي ضئيلة أو فشل المشروع كاملاً. إضافة إلى كافة القضايا التقليدية التي يمكن مواجهتها في المشاريع المجمعة التقليدية، تواجه مشاريع تطوير البرمجيات الموزَّعة المراكز قضايا معينة متعلقة بالتنسيق وحل المشاكل وتطور المتطلبات أثناء التنفيذ ومشاركة المعرفة وتحديد المخاطر. تعتبر أساليب تحديد

المخاطر ومراقبتها في تطوير البرمجيات الموزَّعة المراكز أقلَّ فعالية ويتعين زيادتها. سنناقش ذلك بمزيد من التفصيل لاحقاً.

6-1-2 دورة حياة المخاطرة

في المشاريع التقليدية، يمكن تعريف مراحل المشروع والأنشطة المرتبطة بها كدورة حياة المخاطرة. حدد معهد هندسة البرمجيات دورة حياة المخاطرة (Dorofee [et al.], 1996) بأنها تتضمن الخطوات الآتية:

- **التعريف**: البحث عن المخاطر وتحديدها قبل أن تصبح مشاكل.
- **التحليل**: تحويل بيانات المخاطرة إلى معلومات لاتخاذ القرارات، وتقويم تأثير المخاطرة والاحتمالية والجدول الزمني وتصنيف المخاطر وتحديد أولويات المخاطر.
- **التخطيط**: ترجمة معلومات المخاطرة إلى قرارات واتخاذ الإجراءات التخفيفية (في الوقت الحالي وفي المستقبل) وتنفيذ هذه الإجراءات.
- **المتابعة**: مراقبة مؤشرات المخاطرة واتخاذ الإجراءات التخفيفية.
- **التحكم والسيطرة**: إجراء التصحيحات اللازمة لتعديل الانحرافات عن خطط التخفيف من حدة المخاطر.
- **الإبلاغ**: توفير المعلومات وردود الأفعال الداخلية والخارجية لأنشطة المخاطر والمخاطر الحالية والمخاطر الناشئة.

يتأتى فهم دورة حياة المخاطرة المعرّفة أعلاه من كونها مستمرة. وهذا يعني أنك لا تحدد المخاطرة في بداية المشروع وحسب، بل إنك تحتاج إلى وجود آليات لتحديد وتحليل المخاطر والتخطيط لها ومتابعتها ومراقبتها والإبلاغ عنها باستمرار طوال فترة المشروع. وقد تتغير الأساليب بتطور المشروع. فعلى سبيل المثال، قد تكون

استخدمت طريقة تشخيص أكثر تنظيماً في مراحل مبكرة من دورة حياة المشروع لتحديد المخاطر الكبرى أو الموضوعات المرتبطة بالمشروع والتي لها آليات مختلفة أخف وزناً لتحديد المخاطر الجديدة التي تنشأ بتطور المشروع. سنشرح لاحقاً بعض الأساليب المنظمة المستخدمة لتحديد المخاطر والسيطرة عليها.

6-1-3 المخاطر في سياق مشاريع تطوير البرمجيات الموزَّعة المراكز

تتضمن مشاريع تطوير البرمجيات الموزَّعة المراكز بعض الاعتبارات الخاصة كما ذكر سابقاً، والتي تكون غالباً في صلب القضايا التي اختبرت مسبقاً. نلقي الضوء في ما يأتي على بعض هذه القضايا المهمة.

6-1-3-1 التنسيق

من خلال خبراتنا في مشاريع تطوير البرمجيات الموزَّعة المراكز، نجد العديد من الصعوبات والمشاكل المرتبطة بالتواصل والتنسيق. لا يدرك القائمون على هذه المشاريع مدى أهمية التواصل غير الرسمي (المحادثات التي تجري بين الموظفين أثناء استراحة الغداء أو في الردهات) ودورها الحيوي في تعزيز الجهود المبذولة في عملية البرمجة. أدركت منهجية تطوير البرمجيات السريعة ذلك ووظفته عن طريق مأسسة الممارسات التي تشجع وتعزز التواصل المخصص (كالبرمجة المزدوجة ووجود العميل في موقع البرمجة وقاعات العمل المفتوحة، وغير ذلك). في مشاريع تطوير البرمجيات السريعة، هناك فرق عمل موزَّعة في عدة مناطق جغرافية، وذلك من شأنه أن يجعل هذا النوع من التنسيق صعباً ومكلفاً، حيث لا تتوفَّر العديد من الإشارات الخفية التي تساعد الأشخاص على فهم بعضهم

بعضاً والحصول على الآراء والانطباعات. وتكون النتيجة النهائية لذلك صعوبة توفُّر فهم مشترك لمهام معيّنة بين أفراد هذه الفرق المتباعدة. كما يصعب قياس مقدار الفهم المشترك المكتسب. في مشروع (BAS)، كان هناك عدد كبير من الافتراضات عن المعلومات التي تحتاجها فرق العمل التي تعمل عن بُعد لزيادة الإنتاجية (مثلاً، «ستزودهم بمواصفات البرنامج فقط وهم سيُسرّون بذلك»)، ولم يكن هناك آلية رسمية لتقويم فعالية الآليات التي استخدمت (وثائق التصميم).

أظهر البحث (وخبراتنا تتفق مع هذه النتائج) أن المهام المنفَّذة في مواقع العمل تستغرق وقتاً أطول من المهام التي تنفذ في موقع واحد (في الدراسة، ذكر أنها أطول بمرتين ونصف المرة) (Herbsleb and Grinter, 1999; Herbsleb and Mockus, 2003; Herbsleb [et al.], 2001). إن ما يبدو أنه يؤخذ بالحسبان لهذه الفروقات هو أن هناك زيادة في عملية التنسيق المطلوبة في المشاريع الموزَّعة؛ إضافة إلى ذلك، هناك زيادة في عدد المشاركين اللازمين لإنجاز المهمة في إعداد تطوير البرمجيات الموزَّعة المراكز (Herbsleb and Grinter, 1999; Herbsleb and Mockus, 2003; Herbsleb [et al.], 2001). وهذه الحقيقة غير معترف بها في مشاريع تطوير البرمجيات الموزَّعة المراكز. ويتم عادة تجاهل احتمال وجود صعوبات في عملية التنسيق أو يقلل من شأنها، كما لا يدرس التأثير المحتمل لهذه الصعوبات في المشروع بعمق كافٍ.

إذا نظرنا إلى الدور الذي يلعبه التواصل في تطوير البرمجيات نظرة عن قرب لتحديد سبب صعوبتها في مشاريع تطوير البرمجيات الموزَّعة المراكز، ستبدأ الرؤية في التشكل حول سبب وجود هذه الصعوبات وأنواع الإجراءات التي يمكن اتباعها للتخفيف من حدة

المخاطر. وكما قيل، فإن التواصل غير الرسمي وغير المخطط له أمر مهم في عملية تطوير البرمجيات (Perry [et al.], 1994). تؤدي النقاشات التي تدور بين أعضاء فريق العمل أثناء إعداد القهوة أو حول مائدة الطعام وقت الغداء دوراً مهماً في مشروع البرمجة. إذ تساعد هذه الأحاديث على توثيق العلاقات الشخصية وتنمية مهارات التواصل، وتساعد في إيصال الأنشطة والمهام بين الأفراد المشاركين في المشروع. أثناء هذه النقاشات، قد يُكشف النقاب عن افتراض وجود نزاعات أو تفسيرات خاطئة للأمور أو الروابط بين الأعضاء. كما إنها تساعد في إنشاء شبكة اجتماعية يمكن الإفادة منها للحصول على إجابات للاستفسارات وإيجاد حلول للمشاكل التي قد يواجهها العمل. إن معرفة إلى من يستطيع المرء اللجوء أمر ليس باليسير في المشاريع الكبيرة، لكن وجود شبكة اجتماعية مُمَأسسة جيداً يساهم في الحد من الوقت الذي قد يحتاجه أحدهم في محاولة العثور على الشخص المناسب. وحين العثور على هذا الشخص سيكون أمر الوصول إليه أو إليها والقدرة على التواصل بفاعلية وكفاءة أسهل إذا كان هناك علاقات جيدة معه (Herbsleb and Mockus, 2003).

بانتشار تقنيات معينة مثل الرسائل الفورية وبرامج الدردشة المتنوعة، أصبح أمر نشر التواصل غير الرسمي أسهل من ذي قبل. وهذه الحقيقة تعني أن التواصل غير الرسمي وغير المخطط له صعب للغاية ولا يحدث عادةً بين المواقع. لذا، فإن العثور على الشخص أو الأشخاص المناسبين والتنسيق معهم يستغرقان وقتاً أطول لبيان مدى أهمية العمل خلال مواقع عديدة. لنأخذ المثال الآتي كحالة شائعة تحدث أثناء عمليات الدمج والتكامل:

نحن نؤيد الإصدارات الهندسية الشهرية، التي تتضمن اختبارات للبناء والتكامل. وليس من المستغرب اكتشاف مشاكل أثناء هذه

العملية، وتسجيل وجود خطأ أو خلل في نظام تعقب الأخطاء بالإضافة إلى تعيين شخص أو فريق مسؤول. يقوم «خبير البناء» بتخصيص المهام على أساس الافتراض (أو التخمين) في ما يتعلق بمكان وجود المشكلة. حينها يكون على الشخص الذي تم تعيينه متابعة الموضوع ليكتشف مصدر المشكلة. وقد لا تكون الشيفرة البرمجية التي يتم تعقّبها خاصة بالشخص الذي تم إسناد المهمة إليه، وعادةً ما يكون هناك حاجة للتنسيق مع أشخاص آخرين من مواقع أخرى.

أولاً، ينبغي أن يكون هناك معرفة بسيطة عن الشخص المناسب الذي يجب التواصل به؛ وحينها يبدأ الشخص بالتواصل وترتيب بعض التنسيق المناسب للإجابة عن السؤال. وفي الأغلب يكون لدى الشخص الثاني مهام أخرى ولن يعطي هذه المسألة نفس الأولوية التي ينبغي على الشخص المسؤول أن يوليها إياها (ويصبح هذا الأمر أقرب للواقع عندما يكون الشخصان غريبين عن بعضهما بعضاً)، ما يزيد من مقدار الوقت الذي يستغرقه الحصول على التفاعل الكافي.

وبالإطلاع على هذا الوضع الشائع يتضح كيف أن التنسيق قد يستغرق وقتاً أطول عبر المواقع المختلفة.

يكون عامل التنسيق أكثر حسماً في المشاريع التي لها متطلبات وعناصر تصميم مبهمة أو غير ثابتة (Galbraith, 1997; Kraut and Streeter, 1995) ولكنه يمكن أن يشكل مشكلة في أكثر المشاريع استقراراً أيضاً.

سنناقش لاحقاً الاستراتيجيات المتنوعة للمحاسبة والمراقبة والتقليل من المخاطر المصاحبة للتنسيق والتواصل عبر المواقع.

6-1-3-2 تنسيق الهيكلية

كما تم تناول الأمر في الفصول المتعلقة بهيكلية البرمجيات، يمكن أن تنشأ المشاكل بسبب التنسيق الخاطئ بين الأنظمة قيد البرمجه والمؤسسة التي تقوم ببرمجة تلك الأنظمة. ثمة نهج بسيط خاص بتوزيع العمل يوضح هذه النقطة. فقد تم البدء بمشروع لتطوير برنامج بالغ التعقيد.

كان هذا البرنامج عبارة عن حل مضمّن له قيود مفروضة على الموارد ومتطلبات صعبة حقيقية. إضافة إلى ذلك، كان لهذا البرنامج احتياجات معقدة متنوعة، حيث كان من المفترض أن يدعم مجموعةً واسعة من المنتجات. أثناء عملية التصميم، قام المصممون بعمل رائع يتمحور في تصميم حلٍّ يعتمد على المكوّنات ليتناول المجموعة المركبة من المتطلبات الموجودة في النظام. يتم إسناد الأعمال وتخصيصها بغض النظر عن الحواجز الجغرافية بين الأفراد. فهم يقومون بإسناد الوحدات حسب الوحدة التنظيمية المسؤولة عن تلك الوظيفة. وقد صدف أن تكون تلك الوحدات (وبالتالي الفرق المسندة لتلك الوحدات) موزَّعة عبر خمسة مواقع في عدة بلدان أوروبية.

تتطابق هذه المواقع مع مختلف تملكات في السنوات الأخيرة، وبالتالي ليس من الضروري أن يكون هؤلاء الأشخاص قد عملوا معاً في الماضي. وقد تم تسليم البرنامج في النهاية، ولكنه استغرق وقتاً أطول بنسبة 120 بالمئة من المتوقع، وتجاوزت الميزانية المخصصة بما يقارب 80 بالمئة، ناهيك بمشاكل في الجودة (وهو السبب المباشر للخسارة الفادحة في الأرباح).

يعترف العديد من المشاريع بأن الحدود الجغرافية ينبغي أن تكون مرآة لحدود النظام بطريقة ما، فالأمر أكثر تعقيداً من هذا. يكون الأمر الشائع أن وحدات عمل النظام تحتاج إلى التفاعل بطريقة

ما (لإدراك مجمل متطلبات النظام). وهذا التفاعل يعني نوعاً من التنسيق بين الفرق التي تعمل على برمجة وحدات العمل. السؤال هنا هو: هل يمكن أن نتوقع من هذه الفرق أن تقوم بالتنسيق الكافي لتحقيق هذه التفاعلات كما هو متوقع في المدى الزمني المطلوب؟ في حين أنه لا يمكن الإجابة عن هذا السؤال بشكل ملموس، فسوف تسفر التحقيقات الصريحة عن مجالات غير متطابقة.

6-1-3-3 عدم التأكد والتغيير

يوجد غموض وعدم تأكد بدرجات متفاوتة في كل مشروع. فالمتطلبات والتصميم والمظاهر المختلفة والممارسات الإدارية والعمليات التنظيمية جميعها مجالات قد تكون عرضة لعدم التأكد والغموض والتقلّب. ويمكن أن يشكل عدم الاستقرار في هذه المجالات اضطراباً في أي عملية تطوير برمجيات ـ ولكن أثر هذا الاضطراب يكون أكثر وضوحاً في مشاريع تطوير البرمجيات الموزَّعة المراكز. ويتناول هذا القسم الفرعي باختصار هذه المجالات المتعلقة بالغموض وأثرها في مشاريع تطوير البرمجيات الموزَّعة المراكز. لا يزال يتعين علينا العمل على مشروع مجدول بوتيرة هادئة مع الأخذ بالحسبان كافة القضايا والمهام المحددة في الجدول الزمني وكافة المهام المخصص لها الوقت الكافي لإنهائها (على الرغم من أنه بإمكاننا جميعاً أن نأمل ونحلم). تكون الجداول الزمنية غالباً مضغوطة والفترات الزمنية لا تستوعب كافة المهام والمشاكل المختبرة بشكل كافٍ. وليس من غير المألوف مواصلة المشروع بمهمة زمنية (مثل إشراك فريق بعيد) من دون استكمال كافة الأنشطة المسبقة بشكل كامل. وهذه حقيقة في مشاريع البرمجيات في وقتنا الراهن.

إن عدم التأكد هو أيضاً حقيقة واقعية في مشاريع البرمجيات. ومن المهم تحديد واستيعاب عدم التأكد في كافة المشاريع، ولكنه

أمر حتمي بشكل خاص في مشاريع تطوير البرمجيات الموزَّعة المراكز. وبسبب الطريقة التي تتفاعل بها الفرق والديناميكية التي يمكن أن تتطور بها المشاريع بسرعة، قد نفقد السيطرة على حالات عدم التأكد (الذي قد يكون على شكل مظاهر مبهمة في النظام أو مواصفات مفسرة بشكل خاطئ أو بنود محددة بشكل سيئ) ما يسبب انهيار المشروع. وتظهر خبرتنا في مشاريع تطوير البرمجيات الموزَّعة المراكز أن عدم التأكد يمكن أن يؤدي إلى فرق عمل غير منتجة أو محبطة. ويحدث هذا عندما تضطر الفرق إلى إعادة العمل بشكل جوهري دون أن يكون هناك تقصير من طرفهم أو وضع افتراضات للتقدم. وينطبق هذا الحال أيضاً عندما تكون الفرق عاطلة عن العمل. الأمر المهم هو أنه ينبغي الاعتراف بحالات عدم التأكد صراحة واستيعابها في مشاريع تطوير البرمجيات الموزَّعة المراكز.

6-2 إدارة المخاطر في مشاريع تطوير البرمجيات الموزَّعة المراكز

نبدأ في هذا القسم بمناقشة تفصيلية دقيقة عن كيفية تحديد المخاطر والحد منها في مشاريع تطوير البرمجيات الموزَّعة المراكز. ونصف الأنشطة التي قمنا بتنسيقها بتفصيل أكثر.

6-2-1 تحديد المخاطر

هناك عدة طرق لتحديد المخاطر في المشاريع التقليدية. كما إن هناك مناهج تشخيصية مثل تقويم مخاطر البرمجيات (SRE) (Williams [et al.], 1999) أو ATAM (Kazman [et al.], 2000)، ويمكن لأعضاء الفريق إبلاغ الإدارة عن المخاطر من خلال الاجتماعات الدورية أو تقديم التقارير. جميع هذه الطرق صالحة بنفس الدرجة في مشاريع تطوير البرمجيات الموزَّعة المراكز أيضاً،

ولكنها قد لا تلقي الضوء على المخاطر الخاصة المصاحبة لعملية التنسيق. وتعتمد درجة المخاطر التي يتعرض لها الجدول الزمني والميزانية وجودة المنتجات التي تنبثق عن توزيع بعض التطوير على النظام الذي تقوم ببرمجته، والسياق الذي سيتم تطويره فيه، وسمات الفريق الذي سيقوم بالتطوير. أما الخطوة الأولى في فهم المخاطر الكامنة في مشروع ما، فتتمثل بالتبصر في الشركة والنظر إلى خيارات التوزيع في النظام.

نستخدم منهج التشخيص لتطوير ما يسمى بملف المخاطر لمشاريع تطوير البرمجيات الموزَّعة المراكز. يصف هذا الملف قدرات الشركة التي ستقوم ببرمجة النظام، واحتياجات التنسيق المتضمنة من قبل النظام الذي سيتم برمجته، وتقويم درجة عدم التطابق. يلقي عدم التطابق هذا الضوء على المجالات التي تنطوي على مخاطر عالية. ويكون هذا التقويم مدخلاً لعملية التخطيط والتي تتضمن تحديد الاستراتيجيات المناسبة للحد من درجة التعرض للمخاطر.

6-2-1-1 تحديد القدرة على التنسيق

هناك ثلاثة مكوّنات لطريقة التشخيص هذه: (1) تحليل ترابط النظام (تم وصفه في الفصل الرابع من هذا الكتاب)، (2) تحديد الخصائص التنظيمية، (3) تحليل «التناسق» النسبي بين الاثنين. إن الهدف من تحديد الخصائص التنظيمية هو فهم القدرة على التنسيق بين الفرق ذات الصلة في المشروع. ويتضمن هذا فهم أمور منها:

- الخلفية العامة عن مهارات أفراد الفرق (أو الذين يمكن أن تتشكل منهم الفرق) .
- معرفة الفرق بنطاق العمل المرتبط بالمنتج الذي سيتم تطويره.

- تاريخ التفاعل بين الفرق (مثلاً هل قاموا بالعمل معاً من قبل؟).
- الفصل التنظيمي بين الفرق (مثلاً هل يوجد نفس تسلسل تقديم التقارير؟).
- هل يتشاركون نفس الثقافة واللغة؟

تساهم جميع هذه البنود (وغيرها الكثير) في قدرة الفرق على العمل معاً أو تعيقه. ومن البديهي أن يتم تحديد الفئة التي تنتمي إليها كل خاصية. فعلى سبيل المثال، تكون عملية التنسيق أسهل إذا كان أعضاء الفريق يعرفون بعضهم بعضاً معرفة شخصية. وفي حال أنهم لم يلتقوا من قبل، ستكون العملية أكثر صعوبة. هذا وستتضح الصورة العامة فور تجميع هذه العوامل. هذا ليس علماً دقيقاً، بل هو مصمم لتقديم نظرة حيثما تكمن المشاكل المحتملة والمساعدة على التركيز على أنشطة الرصد.

6-2-1-2 المشاركون

يمكن أن يقوم بهذا النشاط أي شخص يدرك المخاطر المحتملة للمشروع. وباستطاعة المشاركين إجراء مقابلات مع الناس، إن دعت الحاجة، وذلك لجمع معلومات عن المخاطر ومن ثم إعداد تقارير بالنتائج لفريق إدارة المشروع.

6-2-1-3 المدخلات

لإجراء هذا النشاط، ينبغي أن نضع في الاعتبار الأماكن التي يمكن أن تتم فيها البرمجة، ومن سيقوم بتحديد أعضاء فرق العمل. ويمكن أن يساعد هذا التحليل الاختيار بين المواقع، ولكن لا يمكن إجراء هذا النشاط من دون أن يتم اقتراح مرشحين. إضافة إلى ذلك، من المفيد أن يكون هناك معرفة عامة عن النظام الذي سيتم إنتاجه.

كما يجب تحديد أي مجالات معرفة أخرى خاصة كمعالجة الصور. وسيكون من المفيد أيضاً إدراك مجالات الاهتمام بشكل عام (كمواضع السلامة العامة وأداء الزمن الحقيقي، والعمليات المستمرة التي تتم على مدار الساعة).

6-2-1-4 المخرجات

سيكون مخرج هذا النشاط عبارة عن ملف عن إمكانيات التنسيق في شركات البرمجة. وقد يكون هذا الملف رسمياً جداً أو غير رسمي حسب الحاجة، وسيتم ضمه إلى تحليل روابط النظام لتحديد التناسق أو عدم التطابق النسبي، حسب الحالة.

6-2-2 الحد من المخاطر

خلال مرحلة الإعداد، سيكون من الحكمة الحصول على بعض المفاهيم عن ملف المخاطر للشركة كمدخلات. سيتيح ذلك إمكانية التخطيط للمشروع وتوزيع العمل ووضع الممارسات مع المخاطر في الحسبان. هناك أمور ينبغي أخذها بعين الاعتبار للحد من المخاطر. **أولها** البحث عن طرق لتحسين قدرات الشركة؛ و**ثانياً** تخطيط نواحي التنسيق للمشروع؛ و**ثالثاً** التخطيط المسبق لكيفية تعديل المشروع في حال كانت خطة التنسيق الأولية غير كافية (المشروع لا يجري بالسهولة المطلوبة). وسنقوم بمناقشة كل من هذه الأمور بدورها.

6-2-2-1 زيادة قدرات المؤسسة التنظيمية

تذكر أنه وكجزء من المنهج التشخيصي المذكور أعلاه، فإننا نبحث في القدرات الموجودة في الشركة. ففي حال تم تحديد عدم تطابق في التسلسل، يمكن البحث عن وسائل الحد من المخاطر في فسحة الحل (من المحتمل تغيير تصميم النظام) أو التفكير في تغيير

بعض خصائص الشركة. لكل بند تم أخذه في عين الاعتبار، حيث لم تتلقَ الشركة (أو بعض الفرق) تصنيفاً إيجابياً، فكّر في طرق تستطيع من خلالها إحداث تغيير في بعض النواحي في الشركة لتحسين هذا التصنيف. فعلى سبيل المثال، إذا لم تكن الفرق ذات الصلة بالمشروع قد عملت مع بعضها بعضاً في السابق ولا تعرف شيئاً عن أعضاء الفرق الأخرى، حينها يمكن أن تفكر في تبديل الأشخاص أو تنظيم الفرق لبعض الوقت أو التدريب المشترك أو الاجتماعات التنظيمية الدورية أو تمارين بناء الفرق أو غيرها من الآليات لزيادة الوعي وتحفيز الحوار بين أعضاء الفرق الأخرى. على سبيل المثال، يُستخدم اجتماع إطلاق المشروع خلال مشروع تطوير البرمجيات الموزَّعة المراكز لهذه الغاية. ومن الخيارات الأخرى:

- بنية تحتية موحدة (مثال IDE أو نطاق يصطنع التطوير).
- ارتباط أكثر إحكاماً للممارسات الإدارية.
- بناء أكثر تواتراً.
- استعراض ما بين الفرق.

6-2-2-2 وضع خطة للطوارئ

وبما إن الأمور لا تجري دائماً حسب ما هو متوقع، فمن الحكمة التخطيط لهذه الاحتمالية. ويعني هذا في مشاريع تطوير البرمجيات الموزَّعة المراكز، التخطيط للكيفية التي سيتم فيها تعديل عملية تنفيذ المشروع ومكانها. أول شيء سترغب في عمله هو تحديد ووضع أولوية للأبعاد التي يمكنك تغييرها بشكل معقول ضمن شركتك. يمكن أن تتضمن أبعاد التغيير هذه تواتر البناء وتواتر الاجتماعات ونوعها، ويمكن أن تتضمن حتى موقع الفرق. كما ينبغي أن يكون لديك فكرة عن الأوضاع التي ستقوم بتغيير التنسيق وفقاً لها. هذا سيمكنك من منهج أكثر تنظيماً لتنفيذ المشروع. يمكن

أن يساعد هذا المنهج في ضمان الحد من الأثر السلبي لتوزيع المشروع في جودة عدد مرات التسليم ومواعيدها بشيء من التدبر.

6-2-3 رصد المخاطر

تُعتبر عملية رصد التقدم والمخاطر ضمن مشاريع تطوير البرمجيات الموزَّعة المراكز صعبة للغاية. فمن الصعب إدراك ما يحدث داخل الفرق المتباعدة جغرافياً بشكلٍ كافٍ. ففي الوقت الذي تتأخر فيه المنجزات أو يتم اكتشاف مشاكل تتعلق بالجودة، فمن المرجَّح أن يكون الجدول الزمني قد عانى الكثير. وبما إن إعادة التركيز أو التعافي من هذه الحالات يتطلب مزيداً من الجهد والوقت. ومن الطبيعي أن يكون للمطورين فكرة أوضح عن الوضع الحالي للمشروع في أي وقت من تلك التي لدى المديرين. فالخدعة تكمن في جعل المشروع أكثر شفافية والسماح للأشخاص الذين يسمح لهم عملهم بتحديد المخاطر من رفع التقارير بشكل حر.

هناك عدة طرق لإنجاز ذلك، أحدها عقد اجتماعات متكررة حيث يقوم كل عضو في الفريق بالإبلاغ عن الحالة والأمور التي يواجهها، كاجتماعات البدء اليومية. هذه الطريقة فعَّالة، ولكن ينبغي التأقلم معها مع نمو المشروع. فمن الصعب الحصول على انطباع فعَّال بهذه الطريقة عن الحالة العامة للمشروع، للمشاريع الكبرى التي تنطوي على عدة مواقع تطوير.

وقد طورنا أيضاً آليات للأفراد للإبلاغ عن المشاكل التي تواجههم، ولكننا وجدنا أنه من الصعب دفع الأشخاص لعمل ذلك بشكل مسبق. ومن الآليات التي بدأنا باستخدامها وأثبتت نجاحها هي الاستطلاع المنتظم (عادة ما يكون نصف أسبوعي). ويستغرق هذا الاستطلاع عموماً أقل من عشر دقائق لكل شخص، ويطرح أسئلة

عن كيفية إدراك كل شخص لتقدم المشروع، ومع من يتعاملون (وكيف)، وكيف يقضون أوقاتهم. .. إلخ. هذا يمكّن الإدارة من الإطلاع على ملخص البيانات (وبعد بعض الممارسة) تحديد المشاكل بسرعة. بعض المشاكل تكون واضحة على الفور، كالتراجع في معنويات أعضاء الفريق، والأفراد المنهكين في العمل (ويكون هذا الأمر واضحاً عندما يكون هناك قليل من العناصر الأساسية التي تكون مركز العديد من النواحي في المشروع)، عدم التفاعل بين الفرق عندما تتوقع أن يكون هناك تفاعل، وغيرها من الأمور. وهناك مؤشرات أخرى أقل وضوحاً، ولكن يمكن أن تكون مماثلة. وقد وجدنا أنه برصد بعض المؤشرات، تمكنا من اكتشاف المشاكل بشكل مبكر. وتكون هذه المؤشرات أموراً تشبه التذبذب في التواصل غير المخطط (مثال عندما لا تكون مصاحبة لتاريخ إصدار)، وتذبذب غير مخطط في السفر. وقد تمكّنا بفضل الاجتماعات الشهرية للإدارة من إيلاء الاهتمام لهذه المؤشرات، والمراحل المجدوَلة والمخرجات، وقد تمكّنا من اكتشاف بعض المشاكل قبل أن تصبح مشاكل كبيرة، والتحقيق فيها، واستباقها.

6-4 الملخص

تزيد المشاكل المتعلقة بمشاريع، كالتواصل والتنسيق، من المخاطر التي يمكن أن يتعرض لها الجدول الزمني، والميزانية، والجودة مقارنة بالمشاريع المنظمة. وليس من العادة أن تتطلب المهام في مشاريع وقتاً مضاعفاً عن المشاريع المنظمة. ويكون للمشروع فرصة نجاح أكبر في حال تم تحديد المخاطر المحتملة، وتم اتخاذ إجراءات للحد منها أثناء سير عملية التطوير. وقد تم تناول المخاطر وإجراءات الحد منها في خطة إدارة المخاطر.

6-5 أسئلة للمناقشة

1. صِف المخاطر المرتبطة بمشروع تطوير البرمجيات الموزَّعة المراكز، وتعريفها واستراتيجيات الحد منها.

2. صِف كيف يمكنك وضع ملف عن المخاطر.

المراجع

Books

Dorofee, Audrey J. [et al.]. *Continuous Risk Management Guidebook*. Pittsburgh, P. A.: Carnegie Mellon University Software Engineering Institute, 1996.

Galbraith, Jay R. *Organizational Design*. Boston, MA: Addison-Wesley, 1977.

Periodicals

Herbsleb, James D. and Audris Mockus. «An Empirical Study of Speed and Communication in Globally Distributed Software Development.» *IEEE Transactions on Software* Engineering: vol. 29, no. 6, 1-14 June 2003.

Kraut, Robert E. and Lynn A. Streeter. «Coordination in Software Development.» *Communications* of the ACM: vol. 38, no. 3, 1995, pp. 69-81.

Perry, D. E., N. A. Staudenmayer, and L. G. Votta. «People, Organizations, and Process Improvement.» *IEEE Software*: vol. 11, no. 4, 1994, pp. 36-45.

Conferences

Herbsleb, James D. and Rebecca E. Grinter. «Splitting the Organization and Integrating the Code: Conway's Law Revisited.» *Proceedings International Conference on Software Engineering*, 1999, pp. 85-95.

———————— [et al.]. «An Empirical Study of Global Software Development: Distance and Speed.» *Proceedings Interna-*

tional Conference on Software Engineering, 2001, pp. 81-90.

Reports

Kazman, Rick, Mark Klein and Paul Clements. «ATAM: Method for Architecture Evaluation.» Technical Report CMU/ SEI-2000-TR-004, Carnegie Mellon University, 2000.

Williams, Ray C., George J. Pandelios and Sandra G. Behrens. «SRE Method Description (Version 2.0) and SRE Team Members Notebook (Version 2. 0).» Technical Report CMU/ SEI-99-TR-029, Carnegie Mellon University, 1999.



الفصل السابع

عملية وضع خطة المشروع

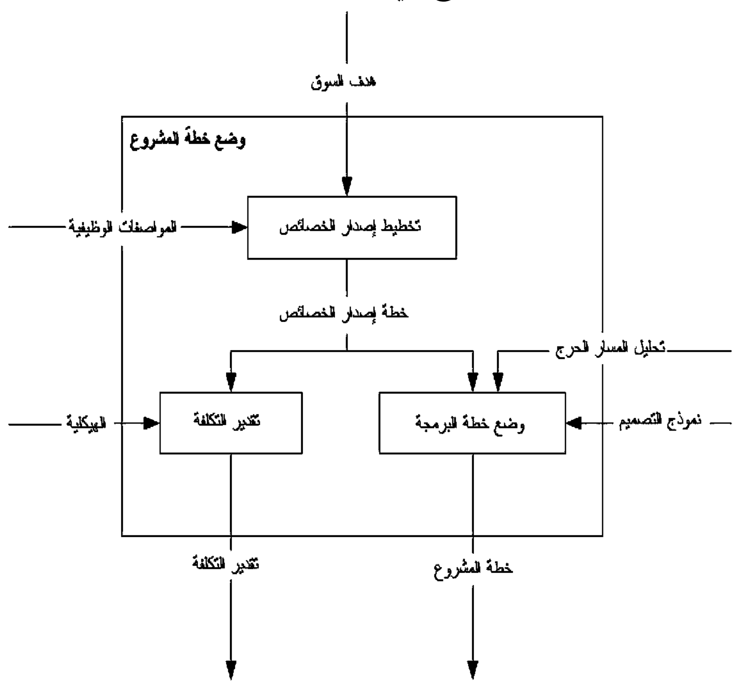
زاد قلق جو عندما قام بالتحقّق من قاعدة بيانات المتطلبات ولاحظ أن عدد المتطلبات، بما فيها طلبات أصحاب المصالح الحالية، قد وصل إلى سبعة آلاف متطلب. وتساءل «كيف سيكون بإمكاننا تنفيذ كل هذه الخصائص، وفي نفس الوقت الالتزام بالموعد المقرر لطرح المنتج في الأسواق؟» يمكن القول إنه قد نجيب عن سؤال جو جزئياً وهو تجميع الخصائص في «إصدارات» أو «نسخ» يمكن تطويرها بشكل متكرّر ومن ثم طرحها في السوق عندما تصبح جاهزة. وباستخدام نموذج التصميم والتحاليل المهمة للمنهج، سيتمكن جو وزملاؤه من تحديد خطة مشروع يمكن أن تتحقق هذه التكرارات من خلالها.

لقد وصلنا إلى قسم إعداد خطة المشروع في هذا الكتاب. نصف في هذا الفصل سير عملية تخطيط المشروع بينما يقدِّم الفصل الثامن تفاصيل عن تقدير التكاليف. سنقوم بذكر كيفية إنشاء خطة مشروع لعدة سنوات. وسنستخدم منهج الجدول الزمني (time-boxed Programe (time) المتكرر لتحقيق برمجة متزامنة في مواقع البرمجة والتطوير العديدة

الموزَّعة في عدة مواقع حول العالم. وبما إن التأخر في تسليم نماذج البرمجيات يحدث حتى في أفضل مشاريع البرمجيات، فسنناقش استراتيجيات التعامل معها في بيئات البرمجة المتزامنة الموزَّعة.

7-1 تخطيط المشروع: لمحة عامة

يقدم الشكل 7-1 لمحة مبسّطة عن الأنشطة المتضمنة في مرحلة التخطيط للمشروع. ففي حين يظهر الرسم الأنشطة على أنها متسلسلة إلى حد كبير، غير أنها لا تكون تكرارية وحسب بل ومقسّمة في مراحل المشروع. ويعتمد تنفيذ هذه الأنشطة على تفاصيل عمليتك الخاصة لدورة حياة البرمجة. ومع هذا سنقوم بتقديم إرشادات حول هذا الموضوع في الفصل الثامن.



الشكل 7-1: مخطط عمل تنفيذ المشروع

تبدأ عملية التخطيط لإصدار خصائص النظام بتعيين «حزم» خصائص النظام التي تشكل الحلول القابلة للبيع. نسمي هذه العملية خطة إصدار خصائص النظام. وحسب احتياجات نظامك ومدى تعقيده، فيمكن أن يكون هذا أداة منفصلة أو لا يكون، ويمكن حينها إعطاء الأولويات لهذه الحزم. وعادة ما يقوم مدير المنتجات في شركتنا بإدارة هذا النشاط.

وحسب الإصدارات المخططة والمحددة في خطة إصدار خصائص النظام والتحليل الوظيفي خلال عملية تعريف الهيكلية (الذي يظهر على شكل نموذج تصميم في الشكل 7-1)، يمكن تحديد النماذج المطلوبة لتحقيق هذه الخصائص. وبتطور التصميم، يمكن تحديد مهام خاصة لفرق البرمجة. وبإمكاننا أن نبدأ بإسناد هذه المهام باستخدام خطة إصدار خصائص النظام والتحاليل المهمة للمنهج (المذكورة في الفصل الخامس من هذا الكتاب). وتعتبر هذه الأنشطة غاية في التعقيد ومقترنة بأنشطة أخرى مثل هندسة المتطلبات وتعريف الهيكلية. ومن غير المحتمل الحصول على معلومات واضحة المعالم قبل البدء بعملية التخطيط، لذا فعادة ما يتم وضع خطة مشاريع أولية رفيعة المستوى وتقدير التكاليف، ومن ثم يتم تنقيحها لاحقاً بشكل متكرر. وسنقوم في الفقرات والفصول القادمة بتقديم مزيد من التفاصيل عن هذه العمليات وطرح أمثلة كذلك من الواقع.

7-2 إعداد خطة إصدار خصائص النظام

يتضمن إعداد خطة إصدار خصائص النظام حزم الميزات في إصدارات خارجية مخطط لها للنظام. تقع مسؤولية هذا النشاط عادة على عاتق مدير المنتج. وتحتمل هذه المسميات الوظيفية أكثر من معنى وتختلف من شركة لأخرى، ولكن المهم أن هذا الشخص

يكون عادة مسؤولاً عن تحديد الخصائص التي ستتيح بيع المنتج في السوق ويكون لديه رؤية في ذلك الشأن. وسيكون هذا النشاط تكرارياً بعض الشيء حيث ستعتمد المجموعة المثلى من الخصائص الأولية على الإطار الزمني أو الجهود المتضمنة في عملية البرمجة وقاعدة العملاء الحالية وحالة المنتجات القديمة المتوفّرة. ويمكن وضع خطة ترحيل للتخلص من المنتجات القديمة بالاشتراك مع خطة إصدار خصائص النظام. وأحياناً، يمكن تجزئة السوق بحيث يكون بالإمكان بيع المنتجات القديمة والجديدة معاً حتى تبني المنتجات الجديدة مجموعة خصائصها. وفي مرحلة زمنية معينة ينبغي تغييب المنتجات القديمة، فلا تحتفظ بها الشركة بعد ذلك.

إن هدف هذا النشاط هو تحديد الحد الأدنى من مجموعة الخصائص التي يمكن بيعها في السوق. ويعرّف هذا الحد الأدنى من مجموعة الخصائص على أنه الإصدار الأول، ويتم حزم الخصائص المتبقية في إصدارات لاحقة.

ويوضح الشكل 7-2 مثالاً على خطة إصدار خصائص النظام، مستنسخة من (Paulish, 2002). في هذا الشكل يكون اسم واضع الجدول الزمني هو اسم المثال على مجموعة الخصائص التي سيتم برمجتها. الميزات المدرجة على القائمة مخصصة لإصدارات متنوعة؛ وحيث تتطور خطة إصدار خصائص النظام، تتحلل هذه الإصدارات إلى إصدارات هندسية داخلية، وهي تتطابق مع سكرم سبرنتس (scrum sprints) (Schwaber, 2001, 2004). هذا، ويعتمد التكامل وخطة الاختبار على خطة إصدار خصائص النظام.

كما يلاحظ من الشكل 7-1، فإن كلاً من هدف السوق والمواصفات الوظيفي هي مدخلات لهذا النشاط. ولبدء العملية ووضع مسودة أولية لخطة إصدار خصائص النظام، ينبغي إحراز تقدم

معقول في أنشطة هندسة المتطلبات وتكون النسخة الأولية من المواصفات التشغيلية متوفّرة. بمعنى أخر، ينبغي أن تكون الخصائص الرئيسة لهذا المنتج محددة (انظر الفصل الثالث من هذا الكتاب). كما إن هناك متطلَّباً مسبقاً آخر لهذا النشاط، ألا وهو هدف السوق الذي يصف أهداف العمل لتطوير هذا المنتج (لماذا تقوم الشركة بصنع هذا المنتج). وهذا يرسِّخ ما تأمل الشركة بتحقيقه حين طرح المنتج، ويمكن أن يكون له تأثير كبير في أولويات الميزات. وحين إعادة تعريف خطط المشروع وخطط الترحيل، يجب إعادة النظر في خطة إصدار خصائص النظام للتأكد مما إِذا كانت مجموعة الخصائص الأصلية المحددة لا تزال منطقية.

	ER1	ER2		ER3		R1	R2+
واضع الجدول الزمني							
البحث في قاعدة المستهلكين عن الأحداث المجدولة	√					√	
إنشاء جدول جديد	√	√		√		√	
التعامل مع التقارير	√					√	
التعامل مع الاكتساب		√				√	
الحصول على أمثل المكتسبات							√
التعامل مع أحداث مجموعة المكوّنات المجدولة		√				√	
عرض الجداول وتحديثها يدوياً				√		√	

الشكل 7-2: مثال مقتبس من خطة إصدار الخصائص ـ برنامج مرحلي

المرجع: بيرسون للتعليم، شركة مسجلة

7-2-1 المشاركون

في حين تقع المسؤولية الأساسية لتطوير خطة إصدار خصائص النظام على مدير المنتجات، غير أن هناك العديد من الأشخاص المشاركين عادة في هذا الجهد. يبين الجدول 7-1 قائمة بالمشاركين، ويسلّط الضوء على مسؤولياتهم الأساسية في ما يتعلق بهذا النشاط.

الجدول 7-1: المشاركون في نشاط تخطيط إصدار خصائص النظام

الوظيفة	المسؤولية
مدير المنتجات	مديرو المنتجات لديهم حق امتلاك هذا النشاط. فهم مسؤولون عن تقديم أو تنظيم مفهوم السوق ومفاهيم عن المنتجات القديمة وعن المنافسة. كما إن عليهم توضيح احتياجات المستخدم النهائي.
مصمم الهيكلية	يشارك مهندس الهيكلية في هذه العملية بالبدء بتوفير المدخلات (كما هي في الوقت الحالي) اللازمة لتنفيذ مجموعة محددة من الخصائص.
مهندس المتطلبات	يوفر مهندس المتطلبات مدخلات في المتطلبات التي تشكل خاصية معينة. وقد يكون ذلك مفيداً لتحديد نطاق مجموعة الخصائص والبدء بالحديث عن الجهود.
مدير المشروع	من المفيد أن يكون هناك مدير مشروع معني بالعملية يمكنه البدء بوضع مسودة جدول لغايات التخطيط.

فكرة مفيدة:

إجبار إدارة المنتج على تحديد الحد الأدنى من مجموعة الخصائص.

من المحتمل أن تواجه إدارة المنتجات وقتاً عصيباً حين تحديد الحد الأدنى من مجموعة الخصائص التي يمكن

بيعها والتفكير في ما يخص الفترات البرمجية التكرارية، حيث سيتم إصدار خصائص نظام إضافية إلى السوق بمرور الزمن. هذا لأن بيع منتج له مجموعة خصائص واسعة متوفّرة في السوق أسهل. قد يتوجب على دائرة البرمجة ممارسة ضغط على إدارة المنتج لوضع أولويات لبعض الخصائص وتأجيل أخرى لإصدارات لاحقة. عادةً ما يكون هناك طلب مثل «أريد كل الميزات على وجه السرعة». وهذا يتطلب إقناع إدارة المنتجات بقيمة الحصول على بعض الخصائص المبكرة ومن ثم بعض الخصائص لاحقاً. ولعمل ذلك يتم حساب أرباح المبيعات المفقودة عن كل أسبوع لا يكون فيه المنتج متوفّراً في السوق.

7-3 تطوير خطة البرمجة

يعطي الشكل 7-3 لمحة عن الأنشطة المتضمنة في عملية تخطيط البرمجة. ولا ينبغي تحديد خصائص النظام على أساس تسلسل البرمجة وحسب، بل ستحتاج إلى تخصيصها لنماذج البرمجيات حسب نموذج تصميم الهيكلية. كما يجب أيضاً تحديد تسلسل برمجة النماذج (انظر إلى تحليل المسار الحرج في الفصل الخامس من هذا الكتاب). وبالتالي ستصبح خصائص النظام متوفّرة ومتكاملة، حيث يتم تطويرها في نماذج فردية تمتلكها الفرق الموزَّعة في عدة مواقع. وحين تتوفّر نماذج كافية وخصائص نظام منفّذة، سيتم تحديد نسخة منتج وتوفيرها لاستخدام العملاء. وفي حال تم تطوير نموذج المتطلبات، كما هو مذكور في الفصل الثالث، يمكن تحديد المتطلبات التفصيلية تلقائياً لكل خاصية أو ميزة. وبالمثل، يمكن اشتقاق النماذج والمناهج المصاحبة اللازمة

لتحقيق كل متطلب تلقائياً من المتطلبات المدمجة ونموذج التصميم.

ومع هذا فليس من السهل تحديد تسلسل لبرمجة وحدة ما. فينبغي تحديد الروابط المختلفة عبر النماذج، وتوفُّر المصادر المناسبة. إضافة إلى المهام المذكورة أعلاه، سيكون هناك أنشطة أخرى في الخطة. على سبيل المثال، في حال كان هناك نواحٍ هيكلية غير واضحة المعالم حالياً، أو وجود مخاطر في ما يتعلّق بأهم المتطلبات من ناحية الهيكلية، قد يكون من الحكمة وضع مفهوم إثبات صحة النموذج بشكل مبكر. وينبغي تعداد هذه المهام وإضافتها إلى الجدول كذلك.

يتم تحديد تواريخ كل مرحلة (والمشار إليها بالإصدارات الهندسية) في الجدول الزمني. وبالتالي يكون على كل فريق تطوير إصدار نسخة تشغيلية من الوحدات لِفرَق الدمج المركزي والاختبار في الموعد المحدد. ويكون لكل فريق نوعٌ من المرونة في ما يتعلق بخصائص النظام التي تم تسليمها. يجب أن يكون الإصدار مستقراً بما فيه الكفاية ليتم دمجه واختباره. ولكن في حال عدم توفُّر بعض خصائص النظام، يتم تخطيط برمجتها خلال المرحلة التالية.

الأهم من ذلك في هذه العملية هو أن تاريخ الإصدار يجب أن يكون ثابتاً. فلا يمكن لأي فريق التغاضي عن تاريخ الإصدار. في بعض الحالات، يمكن لفريق البرمجة عدم المشاركة في الإصدار الهندسي (خلال عطلة محلية مثلاً)، ومع هذا، يتم التغاضي في مثل هذه الحالة عن الإصدار ويقوم الفريق بالإصدار في التاريخ المثبت القادم للمرحلة التالي. المرحلة الحالي مخطط بالتفصيل.

تكون الإصدارات المستقبلية أقل وضوحاً بوجود خطط المشاريع عالية المستوى. فيتم إعادة تخطيط المهام التي لم تنفّذ في المرحلة الحالية لتنفيذ في المرحلة التالية.

فكرة مفيدة:

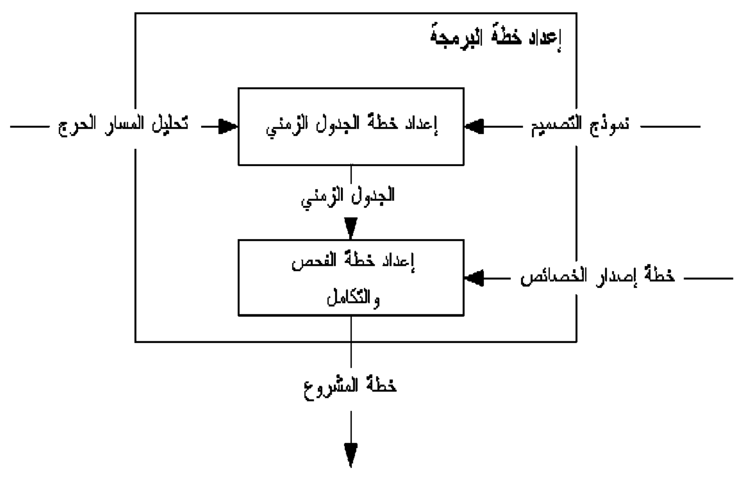
قم باختيار تواريخ مراحل الإصدارات التي يسهل تذكرها.

اختر تواريخ الإصدارات التي يسهل على فريق البرمجة تذكرها ـ على سبيل المثال، آخر جمعة من الشهر. ومع المراحل الشهرية التي تجري كل شهر، يعتاد كل فريق تطوير عمل إصدار هندسي في ذلك اليوم من كل شهر. ينبغي مراعاة العطل المحلية حين تخطيط مشاريع التطوير في عدة مواقع جغرافية في بلدان مختلفة. يمكن تعديل تاريخ الإصدار الاعتيادي بسبب وجود عطلة مهمة (أي في تشرين الثاني/ نوفمبر أو كانون الأول/ ديسمبر). ينبغي أيضاً تتبع إجازات الأعضاء الرئيسين في الفريق. حيث يمكن أن تقوم بعض الفرق بتخطي أحد الإصدارات في حال كانوا يخططون لأخذ إجازة في ذلك الوقت (أي خلال بعض أشهر الصيف في بعض الدول الأوروبية). لذا يجب توخي الحذر من إمكانية قلب بعض المواسم. على سبيل المثال، حين العمل في مواقع في أمريكا الشمالية والجنوبية.

إن إجراء التعديلات على الجداول الزمنية أمر حتمي، حيث تكون التقديرات غير دقيقة أو حين ظهور مشاكل غير متوقعة أو حين وجود فرق دون مستوى الأداء. وتكون الصعوبة عند محاولة الحد من أثر الأنشطة المعاد تخطيطها عبر الفرق.

وبما إن الكثير من المهام مترابطة، فيكون تأخر إحدى الفرق سبباً في إضاعة وقت الفرق الأخرى. وفي كلتا الحالتين، سواء كان الفريق عاطلاً عن العمل أم متخماً بالأعمال، يمكن أن يصبح محروماً من حقوقه بشكل سريع. من الجيد أن يكون هناك خطة طوارئ حيث يكون

بالإمكان نقل المهام في حال كان هناك تباين عن الجدول. لذا يساعد وجود تحليل المسار الحرج المناسب للنظام على تسهيل هذه العملية.



الشكل 7-3: مخطط عمل تنفيذ البرمجة

فكرة مفيدة:

تحديد المهام الطارئة. من المحتمل أن يتم تحديد مهام كبيرة قائمة بذاتها بعد تحديد المسارات الحرجة للمشروع. ينبغي الإشارة إلى هذه المهام صراحةً حيث يمكن نقلها إلى أعلى جدول الأولويات في حال اضطلاع فرق العمل بأعمال أكثر من اللازم أو في حالة تعرضها لأن تصبح عاطلة عن العمل.

7-3-1 المشاركون

يكون الفريق الهندسي مسؤولاً عن تحديد الوحدات وأساليب العمل والمنهجيات اللازمة لتحقيق الخصائص المحددة في خطة

إصدار خصائص النظام وعن تحديد مراحل برمجة هذه الوحدات.

عادة ما يكون هناك الكثير من التنقلات بين مصممي الهيكلية ومديري المشروع أو مديري المنتجات. هذا ويمكن إعادة النظر في خطة إصدار خصائص النظام، ومن المحتمل تعديلها وذلك بتطور الجدول الزمني للمشروع. بعدما يتم تحديد تاريخ الإصدار المقدّر، قد يكون هناك حاجة لتبديل تواريخ الإصدار والتشغيل لإنجاز وتحقيق متطلبات السوق المرغوب بها.

7-4 تقدير التكاليف

خلال مرحلة التحضير، يتم عمل تقدير أولي بالتزامن مع تحديد الهيكلية. نوصي بتفكيك النظام إلى مكوّنات لا تزيد على مئة ألف سطر من الشيفرة البرمجية (100 KLOCs) والتي تحتاج إلى ما لا يزيد على 120 شهر عمل لبرمجتها. نحن ندرك أن هذا تخمين في الأغلب (على الرغم من أنه كلما زادت خبرة مصممي الهيكلية في النطاق المطلوب، كانت التقديرات أفضل)، ولكنها عادة ما تكون خبرة جيدة بما فيه الكفاية لهذه المرحلة، ما يعني إمكانية التوقف إلى هذا الحد في مرحلة التخطيط وتقدير التكاليف. وأما الهدف الرئيس من هذا التدريب فهو إفساح المجال أمام رعاة المشروع لاتخاذ قرار المضي قدماً في المشروع أو التوقف فيه، ولكنه يفيد أيضاً كمسودة لما سيأتي لاحقاً في دورة حياة النظام. وسنقوم بشرح عملية التقدير بمزيد من التفاصيل في الفصل الثامن.

7-5 مرحلة جهود التخطيط

تختلف مراحل المشروع العديدة من شركة إلى أخرى، غير أنه غالباً ما يكون هناك أربع مراحل أساسية بشكل أو بآخر (انظر الفصل

الثاني من هذا الكتاب). تتم عملية إعداد خطة المشروع في المراحل الثلاث التالية بالدرجة الأولى:

1. **مرحلة التحضير (the inception phase)**: يتم في هذه المرحلة الإطلاع على النظام الذي سيتم بناؤه وتحديد ما إذا كان منطقياً ومقبولاً في ما يتعلق بالأعمال. لتحديد مدى جدوى المشروع، يتم أولاً استطلاع متطلبات النظام وهيكلته ووضع خطة رفيعة المستوى وتقدير التكلفة والجهد اللازمين بشكل تقريبي.
2. **مرحلة التطوير (the elaboration phase)**: تتضمن بالدرجة الأولى تحضير هيكلية النظام عن طريق تفصيل المتطلبات وتطوير الهيكلية والمصادقة عليها من خلال بناء شريحة عمودية (هوفمستير وآخرون) وإدخال تحسينات على الخطة.
3. **مرحلة التنفيذ (the construction phase)**: وتتضمن البناء الأولي للنظام حيث تعمل فرق البرمجة بكامل طاقاتها؛ ويتم تسليم المخرجات بصورة دورية؛ كما تتضمن هذه المرحلة تنفيذ عمليات الدمج والتكامل واختبار وظائف النظام ومراقبة التقدم وسير العمل في النظام والتحكم فيه.

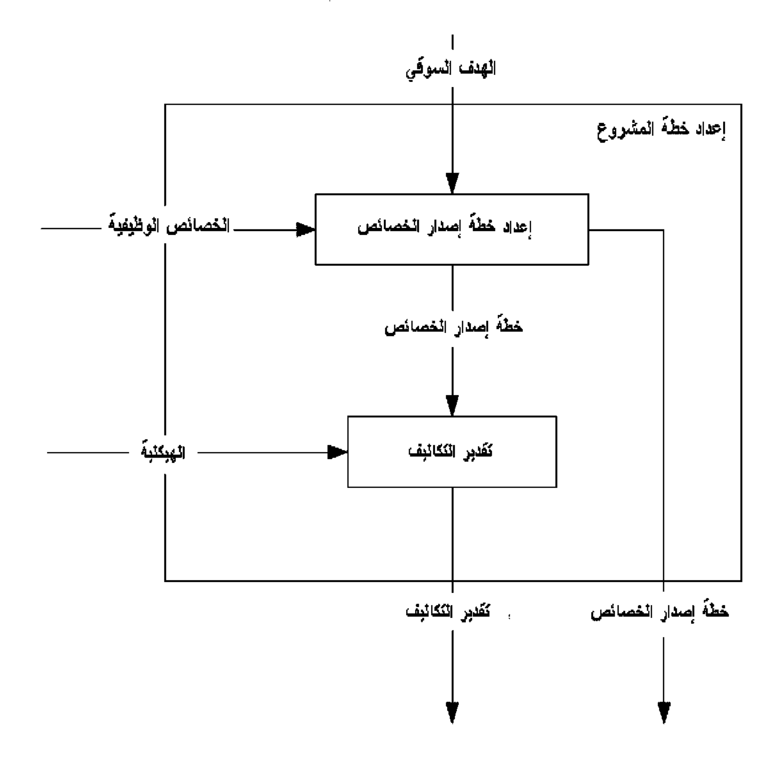
تختلف الاحتياجات من الناحية التخطيطية اعتماداً على مرحلة المشروع والمعلومات المتوفّرة. ويوفر هذا القسم من الكتاب مزيداً من الشرح التفصيلي عن كيفية عمل مراحل لأنشطة التخطيط.

7-5-1 التخطيط خلال مرحلة التحضير

يكمن الهدف الرئيس من وراء مرحلة التحضير في تحديد إذا ما كان المشروع منطقياً بالنسبة إلى الشركة في ما يتعلق بالأعمال أو لا. ولعمل ذلك، ينبغي أن يكون هناك تقدير للتكلفة ذات الصلة والفائدة من بناء النظام. وكما نوقش في الفصل السادس عن عملية التخطيط للمشروع، ويتم خلال هذه المرحلة تحديد المسودة الأولية لخطة إصدار خصائص النظام وتقدير التكاليف. ويعطي الشكل 7-4 لمحة

عن أنشطة التخطيط للمشروع خلال مرحلة التحضير. ومن المهم النظر في البدائل المتوفّرة على الأقل في ما يتعلق بـ:

- الوقت المستغرق لإيصال المنتج إلى السوق.
- وفرة وغنى الخصائص.
- الجودة.
- خطة الترحيل إلى قاعدة العملاء الحالية.
- البنية التحتية المرتبطة اللازمة (أي، الأدوات المرتبطة المطلوبة لبيع النظام بكفاءة كأدوات تحويل البيانات والأدوات الهندسية وأدوات إعداد المخططات والرسوم البيانية وغير ذلك).



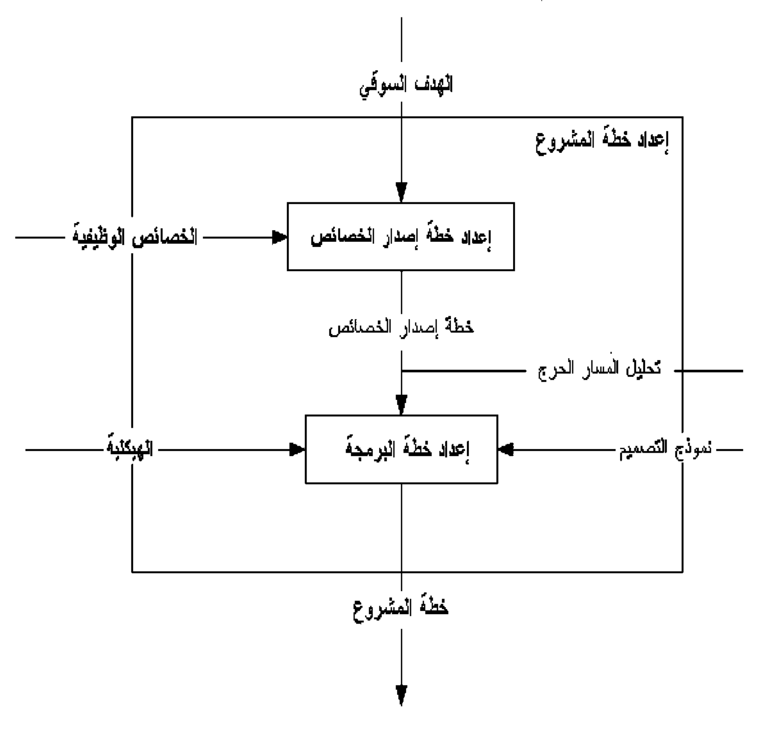
الشكل 7-4: مخطط عمل لمرحلة التحضير

من الممكن تحديد استراتيجية لكيفية تحقيق أهداف العمل. ويجب أن تشتمل هذه الاستراتيجية على إرشادات حول كيفية عمل المبادلات عند مواجهة الصعاب. على سبيل المثال:

لنفترض أن هناك شركة تقوم بإنتاج بضائع لسوق الاستهلاك. دورة حياة هذه السوق قصيرة حيث إن هناك تكنولوجيا حديثة تطرح في السوق كل ستة أشهر، إذ تذهب غالبية المبيعات وأعلى الأرباح للشركة التي تقدم أحدث تكنولوجيا للسوق أولاً. وتقوم هذه الشركة بتطوير منتجات جديدة ومختلفة جذرياً (حيث تشعر الشركة) أن هذه المنتجات ستغيِّر اتجاه السوق. في هذه الحالة، من المهم جداً أن تكون أول الواصلين إلى السوق. ففي حين أن الشركة لن تعترف أبداً بأنها موافقة على التخلص من المنتجات التي تعاني مشاكل في الجودة، عندما يقتضي الأمر، غير أنها تقر أنه من الأفضل طرح المنتج في وقت مبكر في السوق، حتى لو كان يعاني نقصاً في بعض الخصائص المخطط لها وكان يعاني بعض مشاكل الاستقرار. يمكنها بعد ذلك كسب حصة في السوق والحصول على إصدار لاحق، وتحديد القضايا من دون أي تأثير سلبي.

يُتيح وجود هذه الخلفية لدى مدير المنتجات له استخدام هذه التوجيهات عند وضع خطة إصدار خصائص النظام بالإضافة إلى عكس فلسفته على خطة البرمجة التي ستوضع لاحقاً. لقد شهدنا كثيراً قيام المديرين (أو الأفراد) على المستوى المحلي بوضع فرضيات عمَّا هو مهم (على سبيل المثال، لا يمكننا تقليص النطاق، ينبغي علينا تسليم كل شيء!)، واتخاذ قرارات تتناقض مع أهداف العمل الرئيسة للشركة. في هذه المرحلة، يمكن لمديري المنتجات تحديد ما الذي يكوّن المجموعة الأمثل للخصائص الأولية باعتقادهم، وما هو الحد الأدنى لمجموعة الخصائص الأولية، وما الذي بإمكانهم تغييره للتأثير في الحد الأدنى من هذه المجموعات

(على سبيل المثال، إذا قمنا بتغيير السوق الأولي من 'X' إلى 'Y'، حينها يمكن تقديم الخصائص 'أ'، 'ب'، 'ج'؛ أو إذا قمنا بإضافة الخاصية 'Z' إلى منتجنا القديم، حينها يمكننا إطالة عمر ذلك المنتج، وتقليل مجموعة الخصائص الأولية في المنتج الجديد وعرضهما معاً إلى أن يتم إصدار المجموعة الثانية من الخصائص، حينها يمكن تغييب المنتج القديم من السوق). ليس من المهم إنشاء وثيقة رسمية ضخمة، خصوصاً إذا كان هناك قاعدة متطلبات كافية. ولكن، من المهم أن يكون هناك أساس منطقي للقرارات (فضلاً عن البدائل)، إذ قد تكون هذه المعلومات الأساسية مفيدة في وقت لاحق، وعادة ما يتم فقدها.



الشكل 7-5: مخطط عمل لمرحلة التطوير

سيعمل مصممو الهيكلية على تحديد المفاهيم الهيكلية التي ستقود الأنشطة، وسيعملون على تحليل أولي لوظائف النظام بشكل مستقل عن إنشاء خطة إصدار خصائص النظام. يتم عمل تقدير أوَّلي للجهود بالاعتماد على التحليل الوظيفي (كما ورد في الفصل الثامن). ينبغي أن نضع في الاعتبار أهداف العمل حتى تعكس تقديرات التكاليف والجهد أهداف زمن تهيئة المنتجات لتكون متاحة للبيع بشكل مناسب.

7-5-2 التخطيط خلال مرحلة التطوير

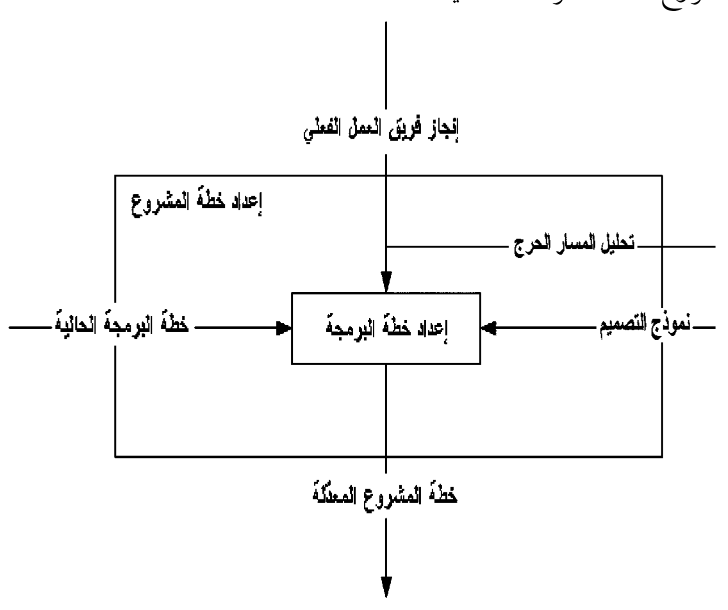
خلال مرحلة التطوير، تصبح المتطلبات والبناء أكثر استقراراً بشكل تدريجي، ما يتيح للمهام أن تكون أكثر تفصيلاً، والتقديرات أكثر دقة، ويتيح تفكيك الإصدارات إلى إصدارات هندسية أصغر، وليصبح الجدول منقَّحاً بشكل أكبر. عادة ما يتم التخطيط بشكل مستمر خلال تنفيذ المشروع. خلال مرحلة التطوير، يتم تخطيط المراحل الأولى بالتفصيل، وما يتبعها من مراحل يتم تخطيطها تدريجياً بشكل أقل تفصيلاً. وبتقدم المشروع، يمكن تعديل الخطط وتخطيط مزيد من المراحل بثقة أكبر. يوضح الشكل 7-5 لمحة عن أنشطة تخطيط المشروع خلال مرحلة التطوير.

7-5-3 التخطيط في أثناء مرحلة التنفيذ

يستمر التخطيط أثناء مرحلة التنفيذ. عند نهاية كل مرحلة، سيكون هناك تقويم نهائي للإطلاع على المواضيع التي سبق اختبارها خلال المرحلة الماضية، والإطلاع أيضاً على التقدم الذي تم إحرازه. وستكون النتيجة تعديل في المهام أو في العملية.

ومن ثم سيكون هناك جلسة تخطيط تفصيلية للمرحلة التالية.

إضافة إلى ذلك، يقوم فريق التخطيط بتنقيح الخطة للمراحل القادمة خلال تنفيذ المرحلة نفسها. ويتم إجراء أهم التعديلات خلال ذلك الوقت. وفي اجتماعات تخطيط ما قبل المرحلة، يسمح فقط بالتعديلات البسيطة. يلخص الشكل 7-6 لمحة عن أنشطة تخطيط المشروع خلال مرحلة التنفيذ.



الشكل 7-6: مخطط عمل لمرحلة التنفيذ

7-6 الملخص والاستنتاجات

قدّمنا في هذا الفصل بعض التوجيهات المتعلقة بتخطيط الإصدارات. ولأننا نستخدم منهج الفترات البرمجية التكرارية، يتم تحليل عملية البرمجة إلى مراحل شهرية صغيرة حيث يتم تطوير خصائص المنتج شيئاً فشيئاً مع مرور الوقت. بالإضافة إلى البرمجة، يتم تخطيط طرح مجموعات الخصائص في نسخ للسوق. أما المهمة

الرئيسة فهي تعريف الحد الأدنى من الخصائص التي يجب أن يحتويها المنتج ليكون جاهزاً للطرح الأول في مواقع العملاء. كما يجب أن يشمل التخطيط على مدة وجود المنتجات القديمة مع المنتجات الجديدة، ومتى وكيف سيتم تغييبها من السوق.

7-7 أسئلة للمناقشة

1. كم يجب أن يكون مقدار الجهد الأقصى الذي يبذله كل فرد من أفراد الفريق؟
2. كم يجب أن يكون الفارق الزمني المناسب بين الإصدارات الهندسية؟
3. ما أهمية تحليل المسار الحرج لتخطيط البرمجة؟
4. كيف يختلف تخطيط المشروع من مرحلة إلى أخرى؟

المراجع

Books

Hofmeister, Christine, Robert Nord and Dilip Soni. *Applied Software Architecture*. Boston; MA: Addison-Wesley, 2000.

Paulish, Daniel J. *Architecture-Centric Software Project Management*. Boston; MA: Addison-Wesley, 2002.

Schwaber, Ken. *Agile Project Management with Scrum*. Redmond, Wash.: Microsoft Press, 2004.

———— and Mike Beedle. *Agile Software Development with Scrum*. 1st Edition. Upper Saddle River, N. J.: Prentice Hall, 2001.

الفصل الثامن

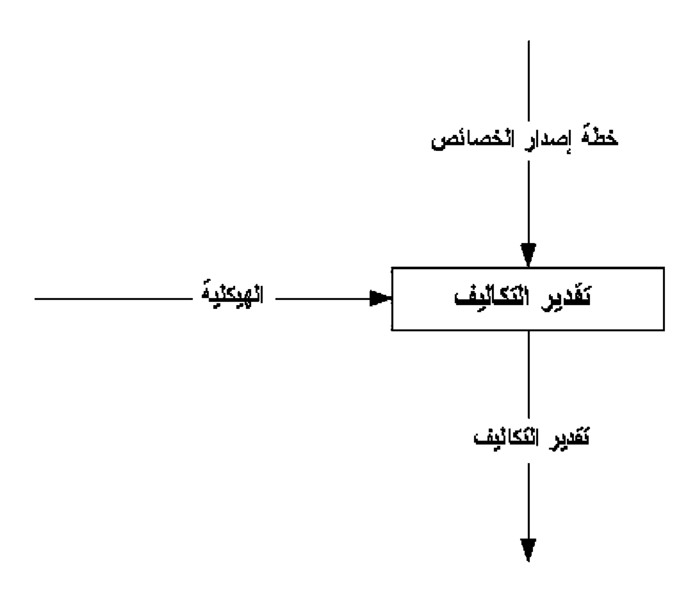
تقدير تكلفة المشروع

دخل مدير بيل إلى مكتبه بعد ظهر أحد أيام الصيف وفي يده علبة من الصودا، وقال لبيل إِنَّه يريد تقدير ميزانية تكلفة المشروع للسنة المالية القادمة بحلول يوم الاثنين. وأضاف: «وبما إنك ستكون قد بلغت مرحلة التنفيذ في ذلك الوقت، أريدك أن تزوِّدني أيضاً بعدد موظفي البرمجة ومتطلبات المكتب وأي تكاليف متعلقة برخص تطوير البرمجيات». وبينما كان يتحدث، رنَّ هاتفه النقال، فلوّح لبيل بسرعة وخرج.

وبينما كان مدير بيل يسير في الرواق، بدأ بيل بالتساؤل عن كيفية تحديد ميزانية تكلفة المشروع لمرحلة التنفيذ. فقد قام بعمل تخمينات لمشاريع من قبل، ولكن كان ذلك باستخدام موظفيه وفي موقعه. أما بالنسبة إلى مشروع تطوير البرمجيات الموزع هذا فسيحتاج إلى تخمين المصادر اللازمة في مواقع برمجة بعيدة، بالإضافة إلى طاقم العمل المحلي. كما إن عليه تخصيص وتوزيع الأعمال بين موقعه والمواقع البعيدة الأخرى. وهو لا يعرف شخصياً فريق العمل

في المواقع البعيدة، وليس لديه أدنى فكرة عن طريقة عملهم وكفاءاتهم.

خلال مراحل البدء والإعداد، كان لدى بيل فريق عمل صغير نسبياً من مهندسي المتطلبات ومهندسي تصميم البرمجيات ممن يعملون لديه. أما بالنسبة إلى مرحلة التنفيذ، فسيكون بيل مسؤولاً عن فريق عمل أكبر بكثير من المبرمجين الموزعين في عدة مناطق جغرافية. وحسب احتياجات زمن تهيئة المنتجات لتكون متاحة للبيع، فكَّر بيل أن جزءاً كبيراً من عملية البرمجة سيتم بشكل موازٍ، وبالتالي سيكون هناك حاجة إلى فريق عمل تطوير كبير نسبياً. وسيكون من الضروري الحصول على هؤلاء المطوّرين الذين سيحتاجون إلى مكان ليعملوا فيه وإلى تدريب وأدوات. ولمّا كان بيل يمعن التفكير في كل هذه الأمور، شعر أن الحرارة ارتفعت قليلاً في مكتبه الدافئ عمّا كانت عليه.



الشكل 8-1: مخطط عمل لتقدير التكاليف

نصف في هذا الفصل منهج تقدير برمجة نماذج البرمجيات في مشروع موزع في عدة مناطق جغرافية، وطريقة التحقق من التخمين الأصلي باستخدام تقنية التخمين من الأسفل إلى الأعلى، وعملية تقدير الجهود للمراحل الشهرية. كما إننا سنناقش تنفيذ وتنقيح التخمينات خلال مراحل المشروع المختلفة. انظر الشكل 8-1، وهو ملخص عن الأنشطة المتضمنة في تقدير المشروع.

8-1 منهج التقدير من الأعلى إلى الأسفل

يقوم التحليل الوظيفي للهيكلية (الفصل الخامس من هذا الكتاب) بتحديد الوحدات الوظيفية للهيكلية التي ستنفِّذها فرق البرمجة الموزَّعة جغرافياً. عندئذ يمكن عمل تقدير لتكلفة البرمجة لمرحلة التنفيذ عن طريق تقدير الجهود للوحدات النمطية التي ينبغي برمجتها خلال الفترات البرمجية التكرارية المتنوعة، وإضافة المصادر اللازمة لوظائف الفريق المركزي (أي، الهيكلية ومهندسي المتطلبات والاختبار وإدارة المشروع). وينبغي إدراك أن مثل هذا التقدير يتم خلال مرحلتي البدء والإعداد. وعادة ما تكون دقة التقدير في مرحلة التحضير 80 بالمئة وتتحسن بحدود 20 بالمئة خلال مرحلة التطوير. أما المدخل الأولي لعملية التقدير فهو حجم تقدير كل وحدة عمل مخصصة، كما يقررها فريق مصممي البرامج.

8-1-1 من هم المضطلعون بهذا النشاط؟

يكون مدير المنتج مسؤولاً عن نشاط التقدير، ولكن يكون لأعضاء الفريق الآخرين دورٌ فيه أيضاً. حيث يقوم مهندسو تصميم البرمجيات بتوفير مدخل لتقدير الحجم، ويمكن أن يشارك مهندسو المتطلبات في التحقق من أن تفاصيل خصائص النظام مأخوذة بعين الاعتبار. يوضح الجدول 8-1 ملخصاً عن المشاركين الرئيسين وأدوارهم.

جدول 8-1 المشاركون في تقويم المشروع

الوظيفة	المسؤولية
مدير المنتج	مسؤول عن نشاط التقويم.
مصمم الهيكلية	يوفر التقديرات المتعلقة بالحجم للوحدات.
مهندس المتطلبات	يقوم مهندس المتطلبات بالتأكد من تحقيق الخصائص المحددة (بمستوى متقدم فقط).

8-1-2 ما هي المدخلات والمخرجات؟

تتضمن المتطلبات المسبقة اللازمة لتقديرات التطوير (تكون هذه التقديرات خلال مرحلة التحضير عبارة عن تخمينات في الأغلب) تحديد الوحدات التي سيتم برمجتها وبعض تقديرات الحجم لهذه الوحدات. وكلما زادت كمية البيانات القديمة المتوفّرة، وزادت ثقة مهندسي التصميم في تقديرات الحجم، تزيد دقة تقديرات التكلفة والجهود. وفي حال لم تتم معايرة البيانات القديمة للمشاريع الموزَّعة في عدة مواقع، يتم وضع جهود إضافية في الحسبان بالاعتماد على التكلفة الإضافية للتعاون في مواقع التطوير الموزَّعة جغرافياً. ويعتمد مدى تأثير هذا التوزيع في المشاريع في خصائص المشروع والشركة. يمكن الاسترشاد بمدخلات أنشطة تحليل المخاطر في هذه التقديرات. وبحكم خبرتنا، وجدنا أن التوزيع عموماً يفرض عامل جهد أكثر بـ 1,2 إلى 2,5 مرة من عامل جهد المشاريع المنظمة.

تكون مخرجات هذا النشاط على النحو الآتي:

- تقدير الجهود المطلوبة من حيث جهود فريق العمل والتوقيت معاً.
- عدد فرق البرمجة وحجمها.

- جهود وحجم أنشطة الفريق المركزي (أي إعداد الهيكلية وهندسة المتطلبات والتكامل والاختبار وإدارة المشروع).

تعتمد إجراءات هذه المخرجات على احتياجات الشركة. كما يمكن تنقيح هذا التقدير عند استقرار المدخلات.

8-1-3 توكيد تطوير البرمجيات الموزَّعة المراكز

لاحظنا أن مهندسي البرمجيات الذين يعملون في فرق برمجة صغيرة نسبياً هم أكثر إنتاجية من المهندسين الذين يعملون في الفرق كبيرة الحجم. وهذا نتيجة حجم التواصل المطرد المطلوب بين أعضاء فريق العمل في الفرق الكبيرة، ما يؤدي إلى مشاكل التزامن المعرفي (أي، عدم تطابق الفهم المشترك) المذكورة لاحقاً في الفصل الثالث عشر من هذا الكتاب.

تم إيجاز توكيد تطوير البرمجيات الموزَّعة المراكز في النقاط أدناه:

- إذا كان حجم وحدات العمل صغيراً بحيث يمكن احتواؤها ضمن إطار عمل الهيكلية وتم تخطيط متطلباتها، فيمكن حينئذ توزيع البرمجة على عدة فرق حول العالم.
- سيكون من المحتم وجود إدارة مركزية لدعم الهيكلية ونموذج متطلبات الأعمال ذات المستوى المتقدم، ودمج النظام والمصادقة عليه، ووضع خطة المشروع، وتصميم واجهة المستخدم رفيعة المستوى، وضمان الجودة، وبرمجة الوحدات المشتركة المهمة .
- إذا تم الإبقاء على حجم فريق برمجة الوحدات صغير نسبياً (مثلاً عشرة أشخاص أو أقل)، يمكن تطبيق عمليات سريعة لزيادة الإنتاجية وخفض وقت البرمجة.

من الناحية العملية، ما يعنيه التوكيد هو أننا نقوم بوضع قيود

على المصممين ليقوموا بتصميم هيكلية تكون فيها وحدات إسناد الأعمال صغيرة نسبياً؛ أي إنه يمكن برمجتها من خلال استثمار جهد عشرة موظفين في العام (أي عشرة أشخاص في اثني عشر شهراً). وهكذا يكون الهيكل التنظيمي الناتج عبارة عن فرق برمجة صغيرة موزَّعة جغرافياً يتم التنسيق بينها من خلال فريق مركزي. تكون الفرق الموزَّعة صغيرة بما فيه الكفاية بحيث يسمح بتواصل جيد بين أعضاء الفريق الواحد يومياً. علاوة على ذلك، يتم تصميم الهيكلية بحيث يكون هناك أدنى حاجة للتواصل بين الفرق (مثلاً الهيكليات المتباعدة). يتم إسناد الأعمال للفرق الموزَّعة بحيث يتم التواصل داخل الفريق ضمن موقع البرمجة أو مع الموقع المركزي فقط، ويتم الحد من التواصل بين فرق برمجة الوحدات في المواقع المختلفة.

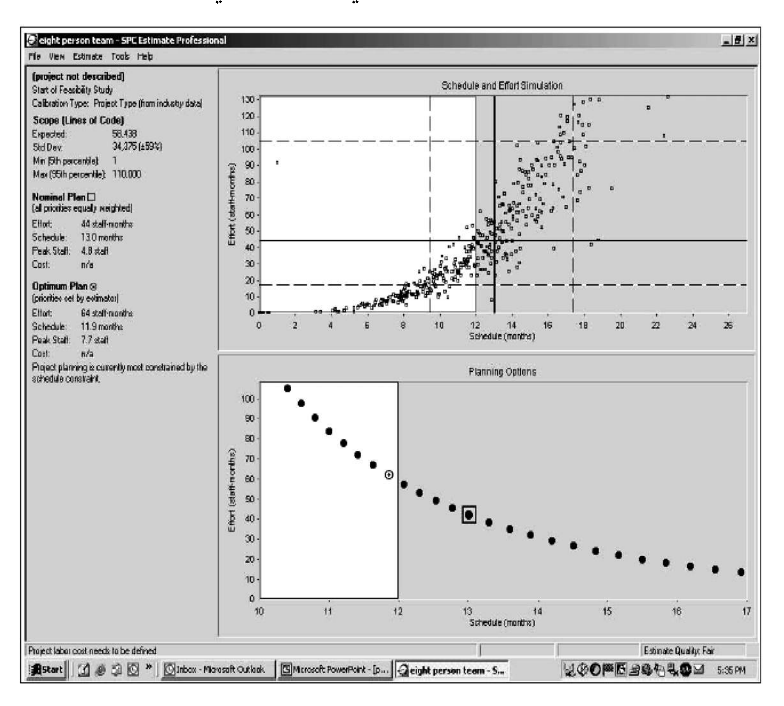
من الممكن مراقبة الأثر السلبي لحجم فرق تطوير البرمجيات المتزايد عن طريق القيام بتحليل «ماذا لو» باستخدام أدوات تقدير التكاليف.

وبينما يزداد حجم شيفرة البرمجة المقدَّرة لمشروع برمجة النظام، يزيد الوقت وعدد الموظفين اللازمين لإنجاز النظام. فإذا ما قام أحدهم بزيادة الجدول الزمني المقترح باستخدام أدوات تقدير التكاليف هذه، ينخفض الجهد وتكلفة البرمجة الناتجة إلى جانب حجم فريق الذروة. مع هذا، يكون هناك ضغط عمل لطرح المنتج في السوق بالسرعة الممكنة.

كذلك فإنه من الأرجح تغيّر المتطلبات والتقنيات خلال المشاريع طويلة الأمد. وبالتالي يكون حجم فريق البرمجة أكبر من الحجم المثالي (من منظور الإنتاجية)، وتكون الفرق الكبيرة أقل إنتاجية بسبب الاحتياج المتزايد للتواصل ما بينها. لذا فإن قاعدتنا الأساسية هي أن لا يزيد حجم أي فريق على عشرة أشخاص، وأن لا يحوي أي موقع برمجة ما يزيد على مئة مهندس (أو عشر فرق

تتكون كل منها من عشرة أشخاص). يكون الحد الأقصى لحجم الفريق اعتباطياً نوعاً ما، ولكنه يسعى إلى أن يتمكن من وضع فريق برمجة الوحدات بالكامل في غرفة اجتماعات واحدة. والوضع المثالي هو أن يتمكن فريق برمجة الوحدات بالكامل من العمل على بعد ما يقارب عشرة أمتار من بعضه بعضاً.

يمكننا توضيح هذه المفاهيم باستخدام أدوات تقدير التكاليف. تأمل مخرجات أدوات التقدير لمشروع يتكون من خمسين ألف سطر من شيفرة البرمجة(*) KLOCs كما هي موضحة في الشكل 8-2.



الشكل 8-2: مقارنة الجهد المطلوب لبدائل مختلفة في الجدول الزمني

(*) KLOC = 1000 lines of code

يتطلب خفض جدول البرمجة ـ لنقل من اثني عشر شهراً إلى عشرة أشهر ـ استثماراً برمجياً أكثر بمرتين تقريباً. هذه المنحنيات غير خطية أبداً، وتفسّر على أنها آثار سلبية المهمة الناتجة من تواصل الفرق كبيرة الحجم. وكمثال على أثر حجم فريق البرمجة والوحدات، يمكننا طرح السؤال التالي: لتطوير نظام برمجيات يقدّر أن يتكون من 100K من النقاط الوظيفية ووجود ثمانمئة مبرمج، فهل من الأفضل تنظيم مئة فريق يتكون كل منها من ثمانية أشخاص أو فريقين، يتكون كل منهما من أربعمئة شخص؟

إذا أدخلنا عدد 1K من النقاط الوظيفية إلى نموذج تقدير التكاليف لواحد من المئة فريق، المكوّن كل فريق من ثمانية أشخاص و50K من النقاط الوظيفية إلى أحد الفريقين المكوّن من أربعمئة شخص حصل على التقديرات الموجزة في الجدول 8-2.

الجدول 8-2: أمثلة على تقديرات إكمال المشروع

	مئة فريق في كل منها ثمانية أشخاص	فريقان في كل منها أربعمئة شخص
جهد (موظف ـ شهر)	64	7118
جدول (شهر)	11,9	59,1
فريق الذروة	7,7	340

بمقارنة هذين الهيكلين التنظيميين يتبيّن، من هذا المثال، أن الفرق المئة الصغيرة يمكنها تطوير المنتجات خلال نصف الوقت والجهد الذي سيحتاجه الفريقان الكبيران. أما الأمر الجدير بالملاحظة فهو الجدول الزمني، إذ تتطلب الفرق الكبيرة ما يقارب خمس سنوات للبرمجة مقارنة بسنة واحدة للفرق الأصغر. لا يأخذ هذا المثال بعين الاعتبار أنه ينبغي دمج وحدات إسناد الأعمال المئة التي

طورتها الفرق الصغيرة أكثر مما يتطلبه الأمر لدمج الوحدتين اللتين طورتهما الفرق الأكبر. نهدف من خلال هذه المناقشة تأكيد الاتجاه بدلاً من التركيز على الأرقام الفعلية؛ إذ ندرك أنه بالإمكان المجادلة في صحة الأرقام المذكورة أعلاه، ولكن الاتجاه يعكس بدقة أن الفرق الصغيرة هي أكثر فائدة من الفرق الكبيرة. هناك طريقة مختلفة للنظر إلى بيانات هذا المثال وهي موضحة في الجدول 8-3.

الجدول 8-3: حجم التعليمات البرمجية مقابل الجهد

KOLCs	الجهد (SM)	الجدول الزمني (أشهر)	فريق الذروة
10	3	5,5	0,9
20	9	7,1	2,0
30	24	8,6	3,8
40	46	10,1	6,3
50	56	10,6	7,5
60	85	11,8	10,3
70	117	12,0	14,6

بازدياد حجم الشيفرة البرمجية المقدرة التي سيتم برمجتها، يزداد الجهد بصورة غير خطية. وبالنسبة إلى مثال الشركة هذه خاصة، تمت معايرة نموذج التقدير لبيانات المشروع السابقة، بما في ذلك العوامل المتعلقة بخبرة فريق العمل ومهاراته وبيئته. والتزاماً بقاعدة العشرة أشخاص والاثني عشر شهراً، يمكننا أن نطلب من المصممين تجنب تحديد أي وحدات نمطية تكون أكبر من ستين ألف سطر شيفرة برمجية.

8-1-4 الحجم

تقوم كافة أدوات تقدير التكاليف باستخدام الحجم المقدّر للمنتج الذي سيتم برمجته كمدخل أساسي. ولسوء الحظ، يمكن أن تكون تقديرات التكاليف التي تنتجها أدوات تقدير التكاليف غير جيدة، وذلك لأن مدخلات تقدير الحجم نفسها قد لا تكون جيدة. من الصعب جداً على المصمم تقدير حجم وحدة البرمجيات قبل برمجتها. وقد تمت مناقشة مختلف المناهج والوحدات لتقدير الحجم في كتب قياسات البرمجيات (Moeller and Paulish, 1993; Paulish, 2002). ونحن لسنا بصدد المشاركة في هذا النقاش هنا، ولكننا نقترح قيام مهندس التصميم باختيار المنهج الأكثر ملاءمة له. أما بالنسبة إلى المشاريع التي تخضع للدراسة والمذكورة في الفصول الأخيرة من هذا الكتاب، فقد قدمت معظم تقديرات باستخدام عدد أسطر الشيفرة البرمجية، وتم ذلك عن طريق عد أسطر الشيفرة البرمجية في مشاريع برمجة سابقة والتي كانت لها وحدات مماثلة للمنتج الجديد قيد التقدير. وهكذا، تكون الخبرة مهمة حين تقدير أحجام الوحدات الجديدة التي سيتم برمجتها.

8-1-5 الجهد

تقدم أدوات تقدير التكاليف مخرجات معلومات فريق العمل حيث يكون الجهد صغيراً في بداية المشروع، ويرتفع إلى الذروة عند نهاية البرمجة، ومن ثم ينخفض مرة أخرى عند وصول المنتج إلى مرحلة الصيانة.

يتم حساب الجهد تحت منحنى ملف معلومات فريق العمل. نحن نقترح استخدام منهج برمجة تكراري ومتزامن حيث يتم برمجة الوحدات بشكل متوازٍ لتحقيق أفضل زمن لتهيئة المنتج ليكون متاحاً للبيع. ويتم تصميم الهيكلية للحد من روابط الوحدات والمحافظة

على صغر حجمها. نتيجةً لعملية البرمجة يمكن البقاء على صغر حجم فرق برمجة الوحدات والحد من حاجتها للتواصل والتواصل عبر المواقع المختلفة.

ونتيجة الطبيعة التكرارية والمتزامنة لمنهجنا في البرمجة، واستخدام المراحل البرمجية لتحقيق تقدم في عملية البرمجة شهرياً، لا نوصي بتراكم فرق برمجة الوحدات في الفترة البرمجية التكرارية. بل نوصي بتوفُّر فريق كامل في بداية فترة البرمجة التكرارية ويبقى معاً بحجم ثابت طوال الفترة التكرارية. من الممكن إضافة فرق أخرى أو تعديل حجم الفرق للفترة التكرارية القادمة. ويمكن لمنهجية تعيين الموظفين السهلة هذه زيادة الجهد الكلي، ولكنها تهدف إلى خفض زمن تهيئة المنتجات لتكون متاحة للبيع وذلك لتحسين زمن تهيئة الفريق والتواصل والكفاءة ضمن فريق برمجة الوحدات. كما إننا نسعى إلى ثبات تعيين فرق البرمجة، وذلك لغايات تراكم الخبرات في المجال عبر الزمن. وهذا يخدم هدفنا بأن تصبح فرق برمجة الوحدات البعيدة بمرور الزمن أكثر شبهاً «بمراكز الاختصاص» التي تتمكن من المحافظة على الوحدات وتعزيزها للفريق المركزي، بدلاً من أن تكون مجرد «مناضد عمل ممتدة».

ستختلف تقديرات الجهود بشكل كبير اعتماداً على من سيقوم بعملية التقدير. ويعود السبب في هذا إلى التنوع الكبير في إنتاجية وخبرة فريق عمل مهندسي البرمجيات. ففي حال قام شخص ما بمهمة مماثلة في الماضي، ستكون احتمالية قيامه بالمهمة الجديدة أسهل بكثير من شخص آخر لم يقم بهذه المهمة من قبل. وينبغي أن يقوم الأشخاص الذين سيؤدون العمل فعلياً بعملية تقدير الجهود. ولكن باستخدام منهج البرمجة المرتبطة بالمواقع الخارجية، فمن الأرجح أن لا يعرف فريق تخطيط المشروع بداهة مستوى خبرة

أعضاء فريق العمل البعيد الذي سيقوم ببرمجة الوحدات. وهكذا ستكون هناك حاجة لنماذج التقدير، ولكن واقع البرمجة سيكون مختلفاً اعتماداً على من سيقوم بالبرمجة. ستتم مناقشة منظور مشاركة الفرق البعيدة في عملية التقدير لاحقاً.

8-1-6 الجدول الزمني

في بيئة العمل اليوم، يعتبر زمن تهيئة المنتجات لتكون متاحة للبيع عبئاً ينبغي على كل مدير مشروع التعامل معه. تحتاج مشاريع تطوير البرمجيات عادةً إلى وقت لتطور ونضج النواحي التقنية. مثالياً، يرغب مهندس البرمجيات في أن يتمكن من التفكير في منهجه الفني قبل تطبيقه، وأن يكون لديه الكثير من الوقت للاختبار وإعادة العمل حتى تتحقق جودة المنتج؛ وهذا يتعارض مع حاجات العمل لطرح المنتج بسرعة، وبالتالي يشعر الكثير من مهندسي البرمجيات بالضغط، إذ إن عليهم تحقيق الجودة العالية ضمن قيود زمنية ضيقة. أما أسوأ سيناريو فهو حين يتم عمل هذا المشروع ضمن جدول زمني غير واقعي، مع إدراك الجميع لذلك، ويبدأ هذا الجدول الزمني بالانهيار. هذا السبب الذي يدعونا للتشديد على الوفاء بكافة مواعيد مراحل المشروع وتقسيم المشروع إلى فترات تكرارية مع تواريخ مراحل شهرية ثابتة حيث يكون هناك إثبات على وجود تقدم في كل شهر. وبتقليص الفترات الزمنية بين مراحل المشروع، يمكن التركيز بشكل أكبر على المشروع، وإذا اقتضت الحاجة، يمكن إعادة التخطيط على نحو أكثر تواتراً.

8-1-7 خطوات التقدير من الأعلى إلى الأسفل

لمساعدة بيل في حل مشكلته المتمثلة في تقدير ميزانية التطوير لمرحلة التنفيذ، سنقوم بوصف منهج تقدير في ثماني خطوات:

1. تحديد أداة تقدير التكاليف. اختر أداة تقدير التكاليف (انظر البند 8-3) وقم بتحديده لشركتك. ويتم هذا بالنظر إلى مشاريع برمجة سابقة وتعديل المعايير أو خصائص التحكم بتكاليف النموذج (Boehm 1981; Boehm [et. al.], 2000) حتى تكون التقديرات المتحصلة من أداة التقدير مقاربة لتكاليف ما قبل البرمجة الفعلية. قم بعمل جدول مشابه للجدول 8-3 باستخدام أداة تقدير التكاليف المحددة.
2. تقدير أحجام الوحدات. جلب تقديرات الأحجام من فريق الهيكلية لكل وحدة برمجيات محددة ضمن الهيكلية. سنستخدم مصطلح «وحدة البرمجيات» بحرية هنا لتعني وحدة إسناد الأعمال التي سيطبقها فريق البرمجة البعيد كجزء من الشيفرة البرمجية الوظيفية التي سوف تدخل ضمن نطاق عمل الهيكلية الذي قام فريق تصميم البرمجيات بتعريفه. ضع قيوداً على فريق مهندسي التصميم حتى لا يكون هناك أكثر من مئة وخمسين وحدة ضمن التصميم، ولا تقدّر أي وحدة بأكثر من مئة ألف سطر من الشيفرة البرمجية (أو ستين KLOCs للشركة المحددة في جدول 8-3). بمساعدة هذه القيود، سيطبّق هذا المنهج فقط على أنظمة البرمجيات المقدرة بخمسة عشر مليون سطر من الشيفرة البرمجية (MLOCs) أو أقل. بالإمكان وضع قيود على مهندسي التصميم ليقوموا بتحديد وحدات أقل من مئة ألف سطر من الشيفرة البرمجية، وبالاعتماد على بيئة البرمجة. إن القيد الأول هو تحديد الوحدات التي سيتم إسنادها إلى الفرق البعيدة التي يمكن أن يبرمجها عشرة أشخاص ضمن فترة برمجة تكرارية مدتها أقل من سنة. وفي حال قام مهندسو التصميم بتقدير وحدة أكبر من القيود المفروضة على الحجم، ينبغي عليهم إعادة تصنيعها إلى وحدات أصغر.
3. إسناد الوحدات إلى الفترات البرمجية التكرارية. ينبغي العمل

عن قرب مع فريق الهيكلية و«خبير الهيكلية» لفهم التسلسل الذي ينبغي برمجة الوحدات بناءً عليه. ستحتاج الوحدات الأكبر إلى وقت برمجة أطول. وقد قمنا بشرح مناهج تخطيط الفترات البرمجية التكرارية في البند 7-2. إذا التزم مصممو الهيكلية بقيود الحجم التي تم تزويدهم بها، لن يزيد الزمن التكراري للفترة الواحدة عن سنة واحدة.

4. تقدير حجم الشيفرة البرمجية لكل فترة برمجية تكرارية. إضافة أحجام كافة الوحدات المسندة لكل فترة برمجية تكرارية. ومن ثم حساب متوسط حجم الوحدة لكل فترة برمجية تكرارية. ينبغي الأخذ بعين الاعتبار أنه يتم استخدام تقديرات الحجم الفعلية لكل وحدة حين وضع خطة البرمجة. ويعتبر استخدام متوسط الحجم لكل فترة برمجية تكرارية في هذه المرحلة أسهل لتحديد تكلفة البرمجة المقدرة لمرحلة التنفيذ بسرعة.

5. تقدير زمن البرمجة والجهد وفريق الذروة لكل فترة برمجية تكرارية. استخدام متوسط حجم الوحدة لكل فترة برمجية تكرارية، حساب زمن البرمجة والجهد وحجم فريق العمل باستخدام أداة تقدير التكاليف المحددة أو من الجدول الذي تم وضعه في خطوة 1 (انظر مثال الجدول 8-3).

6. لكل فترة برمجية تكرارية يتم تقدير زمن البرمجة ومتوسط حجم فريق العمل، وضبط زمن البرمجة إلى أقرب عدد من الأشهر ومتوسط حجم فريق العمل لأقرب عدد فريق ذروة.

7. تقدير زمن جدول البرمجة. حساب زمن جدول البرمجة عن طريق إضافة أوقات البرمجة لكل فترة برمجية تكرارية، ومن ثم إضافة زمن اختبار التحقق في النهاية بالاعتماد على الخبرة المتكونة من المشاريع السابقة (مثلاً: من شهرين إلى ستة أشهر). يمكن اعتبار زمن اختبار التحقق في النهاية مرحلة ثبات.

8. تقدير تكلفة البرمجة. حساب جهد التطوير موظف ـ شهر عن طريق ضرب متوسط حجم الفريق في الفترة البرمجية التكرارية بعدد الوحدات مضافاً إليه جهد المتطلبات المركزية والهيكلية وإدارة المشروع وفرق الاختبار. يتم حساب تقدير تكلفة البرمجة عن طريق ضرب إجمالي عدد الأفراد في سنوات المشروع بمتوسط تكلفة فريق العمل في السنة للشركة. والناتج سيكون عبارة عن التكلفة إذا قامت الشركة بالبرمجة في موقعها. أما إذا رغبت الشركة في برمجة الوحدات في الخارج، فستستخدم تكلفة متوسط فريق العمل في سنوات المشروع لكل شركة مزودة لحساب تكلفة البرمجة، والتي ستكون أقل من التكلفة في موقع الشركة.

أثناء إعداد خطة البرمجة، يؤخذ بعين الاعتبار السيناريوهات المختلفة للبرمجة ومقارنة التكاليف ذات الصلة بالاعتماد على عدد الفترات البرمجية التكرارية والمزودين المختارين وحجم الفريق المركزي وغير ذلك. ويتم إنشاء ملف ذي مستوى متقدم عن معلومات الموظفين وجدول البرمجة الزمني الذي يمكن مراجعته مع الإدارة قبل الالتزام بخطة البرمجة. يبين الشكل 8-3 مثالاً على معلومات فريق العمل.

كمثال قصير على كيفية تقدير تكاليف مرحلة التنفيذ باستخدام الخطوات الثماني المذكورة أعلاه، لنفترض أن لدينا عشرين وحدة تحتاج إلى برمجة، وقمنا بتقسيم العمل على موقعين خارجيين. لنفترض أيضاً أننا قسمنا مشروع البرمجة إلى فترتين تكراريتين تتكون كل منهما من عشر وحدات سيتم برمجتها بواسطة خمس فرق عمل في الموقعين أ و ب. إن حجم الشيفرة البرمجية المقدر للفترتين البرمجيتين التكراريتين الأولى والثانية هو ثلاثمئة ألف سطر وخمسمئة ألف سطر على التوالي. وهكذا يكون متوسط حجم الوحدة ثلاثين

ألف سطر من الشيفرة البرمجية للفترة البرمجية التكرارية الأولى وخمسين ألف سطر من الشيفرة البرمجية للفترة البرمجية التكرارية الثانية. بالاستعانة بالجدول 8-3 يمكن تخطيط الفترة التكرارية الأولى لمدة تسعة أشهر بمتوسط حجم فريق عمل مكوّن من أربعة مهندسين. أما الفترة التكرارية الثانية فسيتم تخطيطها لمدة أحدَ عشر شهراً بمتوسط حجم فريق عمل مكوّن من ثمانية مهندسين.

وهكذا، ستتطلب الفترة التكرارية الأولى ثلاثمئة وستين موظفاً في عدد الأشهر (ثلاثين موظف في السنة) لفرق العمل العشر، أما الفترة التكرارية الثانية ستتطلب ثمانمئة وثمانين موظفاً في عدد الأشهر (73,3 موظف في السنة). ويتم تقدير ذلك خلال مرحلة التنفيذ، وسيكون هناك حاجة لمصممَيْن، ومهندسيْن أو ثلاثة مهندسي متطلبات من مدققي البرمجيات، ومديرَيْ تزويد، إضافة إليك بحكم عملك في موقعك المركزي، وسيكون مجموع ذلك مئتي موظف في أشهر المشروع (16,7 موظف في السنة). وإذا كان تقدير تكلفة الموظف الواحد لمدة سنة في الموقع مئة وخمسين ألف دولار، وفي الموقع 'أ' مئة ألف دولار، وفي الموقع 'ب' ثمانين ألف دولار، فسيكون تقدير التكلفة في مرحلة التنفيذ حوالى 2,5 مليون دولار لموقعك، و5,2 مليون دولار في الموقع 'أ'، و 4,1 مليون دولار في الموقع 'ب'، وبتقدير تكلفة إجمالية تصل إلى اثني عشر مليون دولار. عند العمل على خطة البرمجة، سيكون هناك حاجة للنظر في جدولة وإسناد الوحدات خلال الفترتين البرمجيتين التكراريتين؛ ولكن يمكنك أن تخبر مديرك الآن أن عليه أن يطلب موازنة تصل إلى اثني عشر مليون دولار لأعمال البرمجة خلال عشرين شهراً من بداية مرحلة التنفيذ. في الواقع، ربما ينبغي عليك أن تطلب حوالى اثنين وعشرين مليون دولار، لأن مثل هذا التقدير الأولي يشكّل 80 بالمئة من الهدف المطلوب.

8-2 التقديرات من الأسفل إلى الأعلى

عند تحديد فرق البرمجة المرشحة للعمل، من الحكمة إتاحة الفرصة أمامهم لتقديم تقديرات مستقلة لبرمجة الوحدات. يقوم الفريق المركزي، في محاولة لتحديد الفترات البرمجية التكرارية للإصدار التزايدي، بتقدير من الأعلى إلى الأسفل لوحدة متوسطة الحجم لكل فترة برمجية تكرارية. وسيتم تزويد فرق البرمجة بعدد كامل من الموظفين المتفرّغين من بداية فترة البرمجة التكرارية إلى نهايتها. هذه العملية مفيدة كخطوة تحقق من التقديرات الأصلية، حتى يكون هناك صدقية لفريق البرمجة وحتى تكون ملكية التقدير تابعة له. ولغاية البدء في إسناد مهام البرمجة للوحدات الجديدة من قبل الفريق المركزي، يتم منح فرق برمجة وحدات التطبيق ما يأتي:

- جزءاً من المتطلبات المتكاملة ونموذج التصميم ليتم تطبيقها. لقد تم وصف هذا النموذج باستخدام لغة النمذجة الموحدة (انظر الفصل الثالث من هذا الكتاب). وقد تم إبلاغ فرق البرمجة البعيدة بخصائص التطبيقات بواسطة حالات الاستخدام.
- وصفاً لهيكلية البرمجيات. يتم وصف الهيكلية باستخدام وثيقة وصف الهيكلية أو نموذج التصميم (انظر الفصل الخامس من هذا الكتاب).
- اختبارات القبول. وهي اختبارات القبول التي ينبغي اجتيازها لقبول الوحدة التي تم برمجتها.
- خطة البرمجة التكرارية وتواريخ الدمج. يتم تزويد مخطط جدول زمني محوري يحدد مدة الفترة البرمجية التكرارية والتواريخ الثابتة للتكرارات التي سيتم إصدارها للدمج المركزي وفريق الاختبار.
- مواصفات واجهة الوحدة. تحدد هذه المواصفات كيفية ربط الوحدة التي سيتم برمجتها بإطار عمل الهيكلية.

- تطبيق شريحة النموذج الرأسي. هذا تطبيق خفيف للنموذج للحد الأدنى للتشغيل عبر كافة طبقات الهيكلية المستخدمة كمثال على فرق برمجة الوحدات.
- دليل أسلوب واجهة المستخدم (UI). الذي يحدد تصميم مظهر واجهة المستخدم الذي سيتم استخدامه.

فور تسليم حزمة وثائق التكليف لموقع العمل البعيد، يتم منح فريق البرمجة في ذلك الموقع مهلة مدتها أسبوع للرد عن طريق تقديم تقدير من الأسفل إلى الأعلى لجهود برمجة الوحدات المحددة وفق جدول البرمجة التكراري المحدد. وتشير الخبرة المكتسبة من المشاريع السابقة إلى قاعدة مهمة مفادها أن هناك حاجة إلى أربع ساعات من كل فرد لتقدير جهد الوحدة من الأسفل إلى الأعلى. فلو قام كل عضو في فريق برمجة الوحدات بعمل تقدير وكان حجم الفريق محصوراً بعشرة أشخاص، حينها يمكن تطبيق الأربعين ساعة على تقدير من الأسفل إلى الأعلى لكل وحدة خلال مهلة الأسبوع الممنوحة للرد.

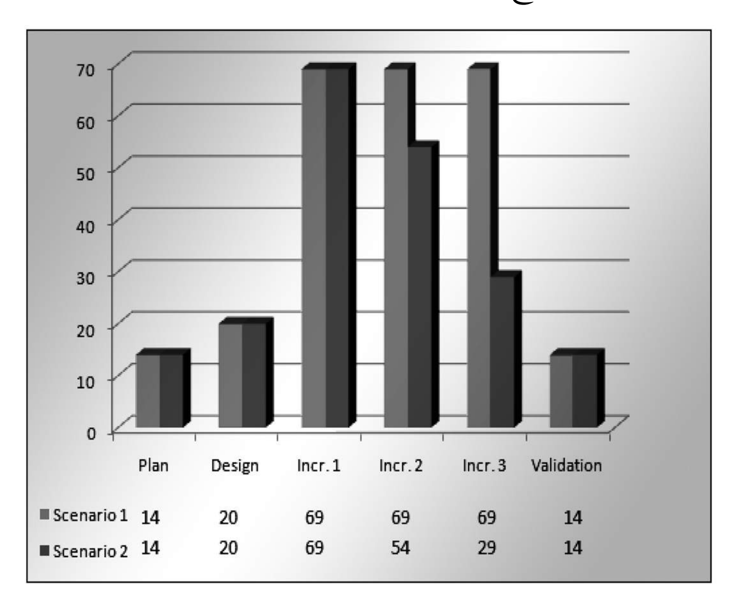
من المقترح أن يقوم الفريق المركزي بقبول تقديرات برمجة الوحدات من دون طرح أي أسئلة تكون ضمن نطاق أكبر بمرتين من تقدير من الأعلى إلى الأسفل الذي تقوم وسيلة التقدير بتقديمه (انظر الجدول 8-3). كما ينبغي التشكيك في التقديرات التي تكون أقل بكثير أو أعلى بكثير من ضعفي التقدير من الأعلى إلى الأسفل. وينبغي مراجعة التقديرات للوحدات التي تحتوي على درجة مخاطرة عالية من المحتمل تعديلها. كما يقترح تزويد كل فريق برمجة بحجم فريق ثابت لكافة مدة الزيادة. يمكن طبعاً الاستعانة ببعض الموظفين الرئيسين لكل فرق الوحدات (كمهندسي التصميم والخبراء واختصاصيي ضبط الجودة ومديري المشروع) في موقع برمجة محدد.

فكرة مفيدة:

لا تقم بإعطاء وعود لإدارتك بعمل توفير كبير في تكاليف البرمجة من خلال القيام بالأعمال في الخارج.
فعلى الرغم من أن أسعار العمالة بالساعة يمكن أن تكون أقل بالنسبة إلى المواقع الخارجية مقارنةً بالموقع المركزي، غير أن محدودية الخبرة في المجال والحاجة إلى التواصل الإضافي قد تعوض أي توفير في التكاليف.
على سبيل المثال، عندما تكون تكلفة العمالة (بالساعة) في المواقع البعيدة ثلث مثيلتها في المواقع المحلية، تحتاج الفرق البعيدة إلى إعادة التطبيقات مرتين إضافيتين للحصول على تشغيل صحيح للمنتج. لذا لن يكون هناك توفير في تكلفة البرمجة. وقد لاحظنا أن الفرق البعيدة قد تحتاج من ستة أشهر إلى اثني عشر شهراً قبل مطابقة إنتاجية الموقع المركزي عند عملهم في مجال عمل جديد أو تكنولوجيا جديدة.

على الرغم من أن نسب تكلفة موظف ما في السنة قد تكون فعلياً أقل في المواقع الخارجية مقارنةً بالموقع المركزي، غير أن تكلفة التواصل الزائدة المصاحبة للعمل في مواقع متعددة ستعوض أي توفير في تكاليف البرمجة؛ وسيحتاج مديرو التزويد في الموقع المركزي للعمل مع الفرق البعيدة، وستزيد موازنات السفر، كما سيكون هناك حاجة لإعادة العمل بسبب الخلل في التواصل. كما ينبغي الأخذ بعين الاعتبار أنه سيتم الاستعانة بمصادر خارجية لبرمجة كميات كبيرة من الشيفرة البرمجية، ولكن يمكن أن تمثِّل جهود وضع الشيفرة البرمجية ما نسبته 20 بالمئة من مجمل تكلفة برمجة المنتج. إضافة إلى ذلك، يمكن أن تكون تكاليف الصيانة المصاحبة للمنتج الناجح مساوية أو قد

تتجاوز تكاليف البرمجة. كما يمكن أن تكون التكاليف المصاحبة لعملية إطلاق منتج جديد في السوق مساوية أو قد تتجاوز تكاليف البرمجة، كأن يتم تدريب موظفي المبيعات. ومع هذا، وعلى الرغم من أن أسعار العمالة في الساعة لبعض الدول قد يثير اهتمام إدارة الشركة، لا بد من الحرص حين أخذها بعين الاعتبار في مجمل تكاليف برمجة المنتج وإصداره وصيانته.



الشكل 8-3: مثال على معلومات الحاجة لموظفين في مراحل المشروع المختلفة

8-3 أدوات التقدير

لقد بدأ استخدام أدوات تقدير تكاليف البرمجيات منذ أن بدأ مديرو العمل في طلب تقديرات عن تكاليف برمجة وتطوير المنتجات البرمجية الجديدة. هناك كتاب يتحدث عن مسألة تقدير تكاليف

البرمجيات وهو «**اقتصاديات هندسة البرمجيات**» لمؤلفه بويهم. يصف المؤلف في هذا الكتاب الذي نشر عام 1981 نموذج تكلفة البناء أو ما يعرف بنموذج (COCOMO)(*). كما تم نشر كتاب **تقدير تكاليف البرمجيات بواسطة** Cocomo II عام 2000. وبالإضافة إلى نموذج Cocomo، تعتمد أدوات تقدير تكاليف البرمجيات شائعة الاستخدام على تحليل SLIM وPRICES وتحليل النقاط الوظيفية (FPA) (Albrecht and Gaffney, 1983; Jones, 1991; Paulish, 2002; Putnam, 1992).

يمكن أن يتراوح سعر أداة تقدير تكاليف البرمجيات من لا شيء (حيث يمكن لأي شخص أن يعثر على وسيلة حساب بسيطة بالاعتماد على COCOMO في موقع عام على شبكة الإنترنت)، إلى آلاف الدولارات (حيث يتم جمع الأداة في عمل استشاري لقياس البرمجيات. إذ يتم تحليل المشاريع السابقة والمستقبلية بالاعتماد على بيانات البيئة والإنتاجية للشركات الحالية). يأتي العديد من الأدوات محمّلاً ببيانات تاريخية من المنتجات المبرمجة لبعض القطاعات الخاصة. وفي حال لم يكن لدى الشركة بيانات تاريخية جيدة من برمجة منتجات سابقة، فبالإمكان مطابقة المنتج بمنتجات مشابهة وتحديد الأداة بالاعتماد على بيانات القطاع. ويقوم العديد من مديري مشاريع البرمجيات في وقتنا الراهن بتحميل أداة تقدير التكاليف على حواسيبهم النقالة حتى يتمكنوا من عمل تحليل «ماذا لو» أثناء ترحالهم.

من الواضح أن الأشكال والرسوم البيانية والجداول التي تنتجها أدوات تقدير التكاليف تثير إعجاب الإدارة غير التقنية. ولكن غالباً ما

COCOMO = Constructive Cost Model (*)

تحتوي هذه الأشكال الجميلة على تقديرات غير دقيقة للغاية عندما تكون مدخلات الحجم غير دقيقة أو عندما لا يتم تحديد الأداة بشكل صحيح. ومع ذلك، تعتبر هذه الأدوات جيدة لمساعدة مديري البرمجة على تذكر كافة الأجزاء التي تدخل في تقدير التكاليف وتقدم لهم إطار عمل زمني كلي (على سبيل المثال، «لا يمكن برمجة هذا المنتج في غضون ثلاثة أشهر؛ أو إذا أمكن ذلك فستكون هذه أول مرة في تاريخ شركتنا»). علاوة على ذلك، يمكن أن يقوم مديرو المشروع باستخدام أداة التقدير والجدول من الأسفل إلى الأعلى لتثقيف الإدارة ووضع توقعات مناسبة حتى يتم أخذ الجداول وحجم الفريق والمخاطر بواقعية في عين الاعتبار قبل البدء في التطوير واسع النطاق في مواقع البرمجة الخارجية.

8-4 الملخص والاستنتاجات

يعتمد منهج تخطيط المشروع والتقويم المذكور في هذا الفصل على التأكيد على أنه يمكن برمجة الكثير من منتجات البرمجيات من خلال فرق برمجة موزَّعة في عدة مناطق حول العالم في حال تمت صياغة المتطلبات وقام مصمم الهيكلية بتقسيم وظائف النظام إلى وحدات صغيرة يمكن تطبيقها تدريجياً.

ويعتمد هذا على الملاحظة التي مفادها أن مشاريع تطوير البرمجيات الأصغر تكون أسهل في الإدارة وتكون فرق البرمجة الأصغر أكثر إنتاجية. إضافة إلى ذلك، يعمل العديد من عمليات تطوير البرمجيات السريعة بشكل أفضل عندما تكون الفرق مكوّنة من عشرة أشخاص أو أقل. مع هذا، يقوم منهجنا في تخطيط المشاريع الموزَّعة باستخدام فريق مركزي لإدارة مجموعات فرق البرمجة الصغيرة التي يشارك كل منها في برمجة وحدة برمجيات تندرج ضمن بناء تصميم البرمجيات.

يمكن تطبيق منهج تقسيم النظام المعقد إلى عدد من الوحدات الصغيرة ضمن تخصصات هندسية أخرى. على سبيل المثال، تم تقسيم تصميم طائرة بوينغ 777 إلى 240 نظاماً فرعياً.

وقد تم تعيين ما يقارب عشرة آلاف موظف في عملية التطوير، ولكن عملت فرق عمل أصغر بكثير على تطوير كل نظام فرعي (Smith and Reinertsen 1997). إن تقسيم المهمة الكبيرة إلى عدد من المهام الصغيرة ضروري لتصميم وتخطيط وهيكلة وتنظيم وتقويم أي مشروع هندسي.

8-5 أسئلة للمناقشة

1. كيف يمكن تقدير أي توفير ممكن في تكاليف البرمجة المرتبطة بالمواقع الخارجية؟
2. ما هي العلاقة بين حجم فريق برمجة الوحدات والسرعة والإنتاجية؟
3. ما هو الزمن اللازم لتعليم فريق عمل يعمل عن بُعد ليصبح منتجاً في مجال عمل جديد؟
4. ما هي الكلفة التي يتم إنفاقها على المهام المتعلقة بالبرمجة لإصدار المنتج وصيانته من إجمالي نفقات برمجة وتطوير المنتج؟

المراجع

Books

Boehm, Barry W. [et al.]. *Software Cost Estimation with Cocomo II*. 1st Edition. Upper Saddle River, N.J.: Prentice Hall, 2000.

———. *Software Engineering Economics*. 1st Edition. Englewood Cliffs, N. J.: Prentice-Hall, 1981.

Jones, Capers. *Applied Software Measurement: Assuring Productivity and Quality*. New York: McGraw-Hill, 1991.

Moeller, Karl-Heinrich and Daniel J. Paulish. *Software Metrics: A Practitioner's Guide to Improved Product Development*. Los Alamitos, CA: IEEE Computer Society Press, 1993.

Paulish, Daniel J. *Architecture-Centric Software Project Management*. Boston; MA: Addison-Wesley, 2002.

Putnam, Lawrence H. *Measures for Excellence: Reliable Software on Time, within Budget*. New York: Prentice Hall, 1992.

Smith, Preston G. and Donald G. Reinertsen. *Developing Products in Half the Time: New Rules, New Tools*. New York: John Wiley & Sons, 1997.

Periodicals

Albrecht, Allan J. and John E. Gaftney. «Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation.» *IEEE Transactions of Software Engineering*: vol. 9, no. 6, 1983, pp. 639-648.

القسم الثالث

الهيكل التنظيمي



الفصـــــل التاسع

فرق تطوير البرمجيات

قام بول في مشروعه الأخير بقيادة مجموعة صغيرة من المبرمجين وفاحصي البرمجيات ممن لهم قدرة على العمل معاً كفريق. وكان قد تم تعيين عدد قليل من أعضاء الفريق حديثاً من الخريجين الجدد. وبعد أن أنجزوا أول إصدار برمجي لأحد العملاء، كان العديد من أعضاء الفريق قد أصبحوا أصدقاء، إضافةً إلى كونهم زملاء. وعلى الرغم من أنه كان عليهم العمل لساعات إضافية أحياناً لتسليم العمل حسب الجداول الزمنية المحددة، غير أنهم كانوا يلتقون بعد العمل لحضور مباراة في لعبة البيسبول صيفاً أو الهوكي شتاءً. استمتع بول بالعمل مع هذا الفريق، واستمتع بقراءة الطُرَف التي كانوا يرسلونها لبعضهم عبر البريد الإلكتروني إلى جانب أداء عمله في مراجعة وقراءة الشيفرة البرمجية التي كانوا يكتبونها.

كان متوقعاً أن يقوم بول في مشروعه الجديد بإدارة مجموعة من فرق البرمجة، وقد كان بعضها موجوداً على مسافات بعيدة عن موقع عمله، وتحديداً في الهند وسلوفاكيا والبرازيل. ولم يقم بتوظيف أو

حتى مقابلة أي من هؤلاء المهندسين البعيدين. بل كان كل ما يعرفه عنهم الأجر الذي سوف يتقاضونه من الشركة عن كل ساعة عمل، وقد كان متحمساً لوقف التعيينات في مكان العمل وجعل جزء كبير من منتجات عمل البرمجة يتم في الخارج. كان بول قلقاً من الآلية التي يمكن بها إنشاء وإدارة ومراقبة هذه الفرق الخارجية. كما إنه شكك في إمكانية مشاركته في أي نشاط في رياضة البيسبول أو الهوكي مع أعضاء هذه الفرق الجديدة.

نوضح في هذا الفصل الهيكل التنظيمي لفرق البرمجة الموزَّعة في عدة مواقع وتحديد قواعد العمل وتحديد المسؤوليات وطرق التواصل والتواصل بين الفرق والتنسيق بينها، وكيفية معالجة أمور المراقبة والتحكم. كما يوضح هذا الفصل بعض آليات العمل المستخدمة لتتبع التطور في المشاريع التي تعمل فيها مثل هذه الفرق.

9-1 هيكلية مشاريع تطوير البرمجيات الموزَّعة المراكز

كما ذكرنا سابقاً، أصبح التنسيق والتحكم والمراقبة أكثر صعوبة بسبب المسافات الشاسعة التي تفصل فرق برمجة التطبيقات بعضها عن بعض. نحن بصدد تقليل الحاجة المفرطة للتواصل عن طريق تحسين التعاون بين المجموعات ما أمكن وذلك عن طريق تقسيم هيكلية النظام البرمجي إلى أنظمة ووحدات أصغر حجماً وأقل ترابطاً. عندئذ يمكن أن يعمل كل فريق على الأنظمة الجزيئية والوحدات بشكل منفصل نسبياً عن الفرق الأخرى. يتم تعريف الوحدات بناءً على الخصائص التي يجب تنفيذها للمنتج وكذلك هيكلية النظام البرمجي كما وصفه المصمم.

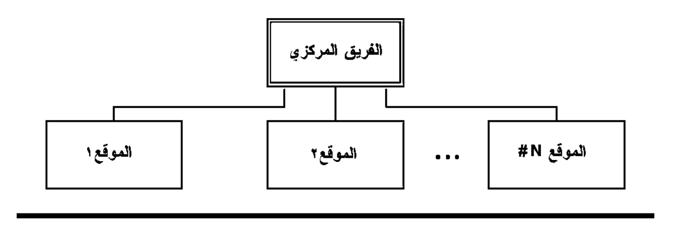
لتحقيق هذا الهدف، يتم تعيين فريق صغير من المصممين كجزء من التنظيم المركزي مع وجود أعضاء يعملون دواماً كاملاً في الموقعين الداخلي والخارجي. ويعمل الفريق كجزء من مشروع،

ويوجد مدير إنتاج مسؤول عن دورة حياة المنتج كاملةً. يقوم مهندسو المتطلبات الموجودون في الموقع المركزي بتحديد وتحليل الخصائص التي يتطلبها أصحاب المشروع.

نُلخّص في الفصل الثامن الأعمال التي يسلمها الفريق المركزي إلى فرق البرمجة التي تعمل عن بُعد لتمكينهم من وضع الخطط وتقدير الجهد الذي يجب بذله لبرمجة الوحدات المطلوبة. يتم إنشاء هذه الأعمال خلال مراحل المشروع المبكرة عندما يكون الفريق المركزي صغيراً ومنتظماً في موقع واحد. نوصي بشدة أن تلقى مهام الفريق المركزي على عاتق بعض أعضاء فرق البرمجة الذين تم تعيينهم بشكل مؤقت في الموقع المركزي وسيتم نقلهم للعمل عن بُعد لاحقاً. يُستغل الوقت الذي يقضونه في الموقع المركزي (على ستة أشهر مثلاً) «لتدريبهم» في مجال التطبيق والهيكلية والأدوات والعمليات التي ستُستخدم أثناء عملية البرمجة. يؤمل أن يصبح هؤلاء الأعضاء قياديين لفرق البرمجة التي تعمل عن بُعد حين عودتهم إلى مواقعهم الأصلية؛ إذ سيبدأون العمل مع مديري التزويد، الذين هم موظفون في الموقع المركزي يساعدون في إدارة العلاقات والمشاريع مع مواقع العمل عن بُعد أو مع مزودي الوحدات التي سيتم برمجتها. لذا، وبشكل عام، تكون النقطة الفاصلة في العملية بين الفريق المركزي المنتظم وفرق البرمجة التي تعمل عن بُعد هي اللحظة التي يتم فيها إصدار التصميم التفصيلي. سوف يبقى الفريق المركزي مجتمعاً خلال سير عمليات البرمجة، لكن قد يعود بعض أعضاء الفريق إلى مواقعهم الأصلية.

خفّت حدة بعض مخاوف بول في ما يتعلق بالعمل مع مصادر غير معروفة بفضل انتقال الأعضاء بين الموقع المركزي وموقع التزويد. وقد لا يعرف بول الجميع في مواقع البرمجة التي تعمل عن بُعد، لكن لا بد أنه قد اكتسب خبرة على مدى ستة أشهر عمل

أثناءها مع بعض الأشخاص الأساسيين في الموقع البعيد والذين قد عملوا معه في الموقع المركزي. نعتقد أن هذا النوع من التواصل الشخصي ضروري لدعم التواصل مع مواقع العمل البعيدة خلال عملية البرمجة. وقد يكون من المهم بشكل خاص معرفة الزملاء في موقع التزويد بشكل شخصي وعميق حين حدوث أي مشكلة في المشروع. ويكون حل المشاكل عن بُعد صعباً إذا لم يكن أعضاء المواقع المركزية والبعيدة قد التقوا أو عملوا مع بعضهم بعضاً.



الشكل 9-1: مثال على تنظيم الهيكلة التنظيمية لعملة البرمجة الموزَّعة المراكز

وكما ورد في الفصل الثاني، يقوم الفريق المركزي المنتظم بتعديل فريق عمل المشروع خلال مرحلتي البدء والتطوير. حالما تبدأ مرحلة التنفيذ، يتم إضافة فرق البرمجة التي تعمل عن بُعد إلى المشروع، لذلك يكبر حجم الفريق الكلي كمجموعة من الفرق غير المترابطة. يتخذ التنظيم الكلي للمشروع الهيئة الموضحة في الشكل 9-1، إذ يعمل الأعضاء الموجودون جميعهم في المواقع المختلفة على المشروع.

تستمر وظائف الفريق المركزي التي يتم تنظيمها أولاً خلال مرحلتي البدء والتطوير وحتى الوصول إلى مرحلة التنفيذ. يقوم الفريق المركزي بقيادة الفرق التي تعمل عن بُعد ومراقبة عملها. يقوم الفريق

المركزي بإنجاز التصميم المتعلق بالمفاهيم وهندسة المتطلبات، يتم تصميم وبرمجة الوحدات بواسطة الفرق التي تعمل عن بُعد، ومن ثم يتم دمج واختبار المنتج ضمن فترات برمجية تكرارية طالما تقوم فرق البرمجة عن بُعد بتسليم الشيفرة البرمجية إلى الفريق المركزي. لذلك، يكون الفريق المركزي هو مركز السيطرة والتواصل والذي يجعل تطوير البرمجيات الموزَّعة المراكز أمراً ممكناً. تتضمن الوظائف التي يؤديها الفريق المركزي ما يأتي:

- **هندسة المتطلبات.** تحليل المتطلبات التي يطلبها العملاء أصحاب المشروع. وتوزيع خصائص النظام إلى إصدارات ووحدات. تطوير نموذج المتطلبات. وتوفير الخبرات في مجال العمل المطلوب في فرق البرمجة.
- **تصميم هيكلية البرمجية.** تصميم ووصف الهيكلية من حيث المفاهيم. إنشاء التصاميم ذات المستويات المتقدمة (يمكن ذلك بمساعدة الفرق التي تعمل عن بُعد). مراجعة ومراقبة التصاميم حين تطبيقها. كما يمكن أن يطوِّر فريق تصميم هيكلية التطبيقات نماذج لاختبار هيكلية التبادل وتحديد الجدوى التقنية منها. وقد يتم إحضار هذه النماذج من الموقع المركزي أو من المواقع التي تعمل عن بُعد. هذا وسوف يحدد المصممون التقنيات والأدوات والإجراءات التي سيتم استخدامها في البرمجة.
- **وضع خطة المشروع.** وضع التقديرات. إعداد الجدول الزمني والموازنات والإجراءات. واختيار مواقع البرمجة التي تعمل عن بُعد وتحديد مهامها. وتحديد الأهداف الفورية.
- **تصميم واجهة المستخدم.** تصميم دليل لنمط واجهة المستخدم. وتطبيق تحليل الاستخدام والاختبار. وإعداد نموذج لمخطط سير واجهة المستخدم.

- **تكامل النظام والتحقق من صحته**. إعداد خطة للإنتاج. وإعداد نموذج اختبارات قبول الوحدات. وإعداد اختبارات النظام وإجراء الاختبار خلال عملية البرمجة التدريجية.
- **ضمان الجودة.** تحديد عمليات الجودة والاختبارات. والمساعدة في تحليل مخاطر المشروع. ومراجعة عمليات وانجازات موقع التزويد. والمساعدة في تحديد عمليات المراجعة. المساعدة في تطوير عمليات البرمجة.

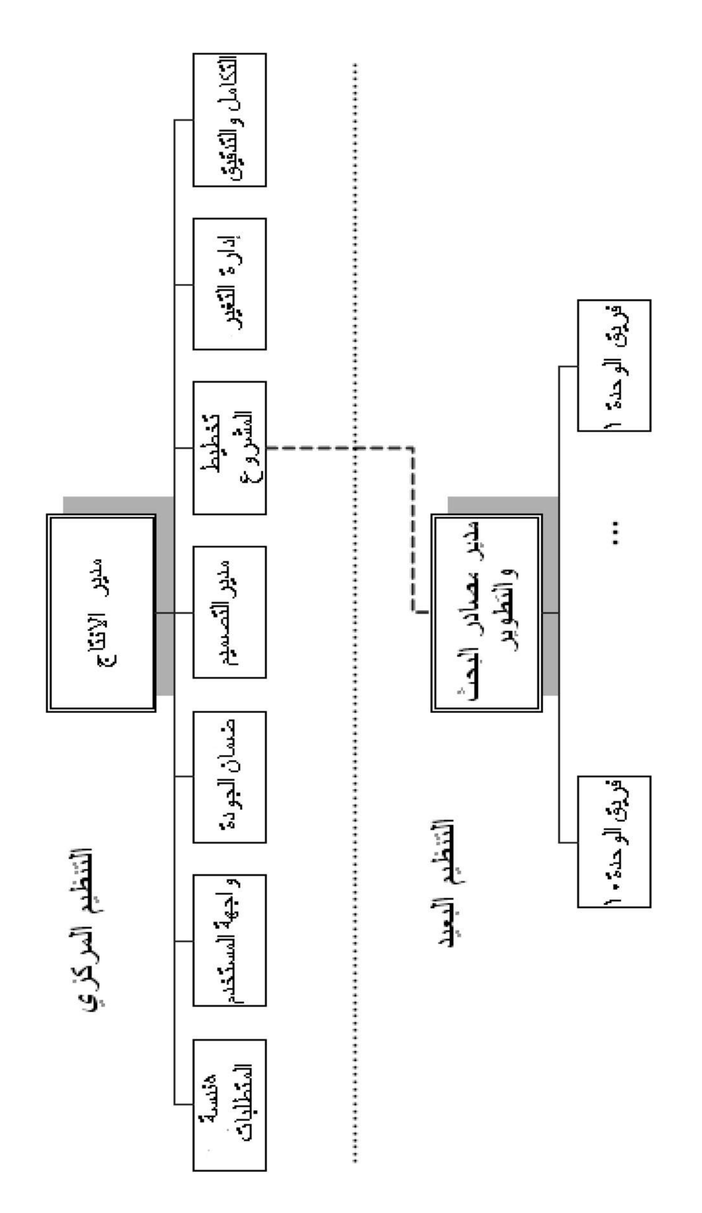
يتم نقل حزمة التوثيق من الفريق المركزي إلى جميع فرق برمجة الوحدات التي تعمل عن بُعد كما وُصف في الفصل الثامن. يتم استخدام حزمة التوثيق هذه للمساعدة في نقل العمل الذي سيتم إنجازه إلى فرق البرمجة التي تعمل عن بُعد وفقاً لخطة البرمجة. ويتم تحديد نطاق العمل المطلوب تنفيذه من قبل فريق برمجة يعمل عن بُعد صغير الحجم نسبياً (عشرة أفراد كحد أقصى). يتم وصف وظائف أعضاء هذه الفرق لاحقاً. لكن وبشكل عام، لا بد أن يمتلك هؤلاء الأعضاء مهارات متعددة بما في ذلك مجال العمل والتصميم والبرمجة والاختبار. إضافة إلى ذلك، سيتم ربط فرق البرمجة التي تعمل عن بُعد مع الفريق المركزي من خلال مدير التزويد بشكل أساسي وأيضاً مع وظائف الفريق المركزي التي تم ذكرها سابقاً كتصميم التطبيقات وتكامل النظام واختباره. في الواقع، سيتم تشكيل مجموعة مركزية لكل وحدة بحيث تعمل الفرق البعيدة بخبرات متعددة من الفريق المركزي. فعلى سبيل المثال، إذا كان فريق البرمجة (A) الموجود في الهند يعمل لإنجاز إطار عمل وحدة واجهة تطبيق المستخدم، قد يعمل في مجموعتهم المركزية كلٌ من بيل الذي وضع المواصفات الوظيفية للوحدة، وجورج من فريق التصميم، وآرون من فريق تصميم واجهة المستخدم، ودان مدير التزويد، وغيرهم.

بناءً على ما سبق، سيتكوّن التنظيم الكلي للمشروع من فرق برمجة تعمل عن بُعد تتناسب مع الهيكل التنظيمي للشركة المحلية التي تقوم بإرسال تقارير إلى الفريق المركزي المكوّن من خبراء تقنيين، كما تم وصفه سابقاً. تزوِّد الفرق التي تعمل عن بُعد الفريق المركزي بشيفرات البرمجة وأي أعمال أخرى لازمة، ويقوم الفريق المركزي باستخدامها لبناء المنتج. على كل حال، نوصي أن يتم إنشاء هذه العلاقة من المزود على المدى الطويل. لذلك، فمع مرور الوقت، سوف يقوم التنظيم الذي يعمل عن بُعد ببرمجة تطبيقات أكثر للتنظيم المركزي، ويصبح ذا علاقة بأنشطة المراحل الأولية بشكل مطّرد. إضافة إلى ذلك، ولنجاح مثل هذه العلاقة بمرور الوقت، نوصي بتبادل أعضاء فرق العمل كمندوبين لفترات طويلة أو قصيرة بين الفرق التي تعمل عن بُعد والفريق المركزي. وسيعتمد نجاح أي شركة، وبخاصة تلك التي تعمل بطريقة تتخطى الحدود الوطنية والجغرافية، على مدى الثقة الموجودة بين الزملاء والعلاقات بينهم. ويمكن أن تتطور هذه الثقة المتبادلة بين موظفي الشركة بتطور التفاعل الشخصي بينهم. في الوضع الحالي لممارسات هندسة البرمجيات، نشك في أن يكون أسلوب الاستعانة بالمصادر الخارجية المتبع ناجحاً حيث يتم إرسال المواصفات إلى مواقع العمل عن بُعد، ومن ثم انتظار وصول شيفرة البرمجة (Herbsleb [et al.], 2005).

يوضح الشكل 9-2 مثالاً لشركة مبيناً فيه العلاقة بين الفرق المركزية والفرق التي تعمل عن بُعد. ويتحمل مدير المنتج المسؤولية الكاملة عن الفترة اللازمة لبرمجة المنتج. ويكون رئيس التصميم مسؤولاً عن أعضاء فريق التصميم ويتحمل المسؤولية عن أي قرارات تقنية قد تؤثِّر في المشروع. يقوم أعضاء الفرق التي تعمل عن بُعد في

برمجة الوحدات بإرسال تقاريرهم إلى مدير البحث والتطوير الذي يوجد في مواقعهم. وترسل الفرق التي تعمل عن بُعد التقارير إلى مدير المشروع في الموقع المركزي من خلال مدير التزويد المحدد لهم. تقوم المجموعة المركزية لكل وحدة بربط الاختصاصات الوظيفية المختلفة بين الموقع المركزي ومواقع العمل عن بُعد.

المثال الموضح في الشكل 9-2 هو مثال على العلاقة بين الفريق المركزي وأحد المواقع التي تعمل عن بُعد. ويجب أن يتم تنظيم الوحدات والتطبيقات المتشابهة ما أمكن. ويتم تجميع الوحدات إلى تطبيقات أو أنظمة جزئية ويتم تكليف فرق البرمجة بها في موقع واحد، بحيث لا يتعدى عدد المبرمجين في الموقع مئة مبرمج. ويرتبط الهيكل التنظيمي عادةً في مواقع العمل عن بُعد مع الموقع المركزي. على كل حال، يكون التواصل بين عدة مواقع عمل بعيدة ضرورياً أحياناً، كأن يكون هناك وحدتان تتطلبان واجهة تطبيق مستخدم يقوم بإنشائها كلا الفريقين. من المفيد جداً أيضاً أن تتبادل المواقع الخبرات التدريبية المكتسبة. ويكون هذا مفيداً حين إحضار فرق جديدة أو استبدال فريق بآخر بحيث تتعلم هذه الفرق عن الأنظمة من الفرق الأخرى. وفي كل الأحوال، يجب أن يتم تكوين هذا الارتباط بحذر مع مرور الوقت، لأنه ربما ينظر كل فريق إلى الفريق الآخر بشكل تنافسي في بداية الأمر إلى أن يتم تكوين مجموعة من الخبرات وبناء ثقة متبادلة.



الشكل 9-2: العلاقة بين الفرق المركزية والفرق التي تعمل عن بُعد

فكرة مفيدة:
يجب أن تقاوم أي اقتراحات قد تحوّل الفريق المركزي إلى فريق وهمي خلال مرحلتي البدء والتطوير.
لاحظنا أن بعض الأنشطة تتطلب تفاعلاً مستمراً ومتقارباً بين أعضاء الفريق كتحليل المتطلبات وتصميم الهيكلية. لذلك نؤكد أن مثل هذه الأنشطة يجب أن تكون منتظمة.
نظراً إلى الحاجة إلى إحضار أعضاء جدد للفرق التي تعمل عن بُعد خلال المراحل الأولية للمشروع، ستطرأ أمور لوجستية. عندما يتم نقل أعضاء من فريق يعمل عن بُعد ليعملوا في موقعك، سوف تزداد التكلفة بسبب تكاليف السفر والمعيشة؛ إن تغيير موقع عمل أعضاء الفريق وابتعادهم عن عائلاتهم يؤدي إلى الشعور بالغربة أو قد يؤدي إلى رحلات عودة غير متوقعة. ويستطيع أعضاء الفرق التي تعمل عن بُعد العمل في الموقع المركزي برفقة عائلاتهم. وإذا لم يكن ذلك ممكناً، يتم التخطيط لرحلات منتظمة بحيث يعمل الفريق المركزي لعدة أسابيع، ومن ثم يعود الجميع إلى بيوتهم لمدة أسبوعين. على كل حال، كان لنا تجارب سيئة عندما حاولنا في إدارة فريق مركزي لا يوجد في موقع واحد.

9-1-1 الوظائف والمسؤوليات

يلخص الجدول 9-1 وظائف ومسؤوليات الفريق المركزي، بينما يلخص الجدول 9-2 وظائف ومسؤوليات الفريق الذي يعمل عن بُعد.

الجدول 9-1: وظائف ومسؤوليات الفريق المركزي

الوظيفة	المسؤولية
مدير المنتج	يمتلك المنتج الذي يحدد متطلبات السوق؛ يقوم بوضع خطة إصدار المنتج ويحدد مجموعة الخصائص التي يجب أن تكون جزءاً من المنتج.
مدير المشروع	يضع خططاً لدورة حياة المنتج، يقوم بإدارة فرق المتطلبات والتصميم المتعلقة بخطة المشروع وتكليف عناصر فريق البرمجة الموزَّعة عالمياً.
مدير التزويد	يضع خططاً لدورة حياة أحد عناصر المشروع، ويعمل على إدارة أحد عناصر فرق البرمجة التي تعمل عن بُعد.
الخبير في مجال العمل	يوفر الخبرات الضرورية للعمل، ويراجع مهارات متطلبات العمل ويبين المميز منها.
مهندس المتطلبات	يجمع طلبات العميل. يضع سياسة لتدقيق طلبات العميل وتقسيمها إلى متطلبات، يطور نموذج التحليل وقائمة المصطلحات. يترجم النموذج التحليلي إلى مستندات تتضمن متطلبات العمل.
مصمم الهيكلية	يقوم بتعريف ووصف وربط هيكلية البرنامج. ويتخذ القرارات التقنية. ويطبق النموذج التصميمي. وهو مسؤول عن نجاح أحد جزئيات المكوّنات. يطور المواصفات واختبارات القبول لمكوّنات المنتج.
مسؤول تطبيق العمليات (خبير في ضمان الجودة)	يقوم بدعم العمليات والتدريب على مهام متنوعة. ويدعم الإدارة المؤتمتة لإنشاء وتهيئة واختبار الوحدات ومتابعة أخطاء النظام بالاستعانة بفريق هيكلية المشروع. وإدارة أخطاء النظام واختبارها. مسؤول عن دعم البرمجة وعمليات الاختبار وأدوات ضمان الجودة. مسؤول عن ضمان القدرة على تتبع نتائج المشروع.
مصمم واجهة المستخدم	يطور مواصفات واجهة المستخدم، إنشاء نماذج الشاشات، إعداد نماذج مبدئية لواجهة المستخدم، واختبار واجهات المستخدم التي يتم تسليمها.

ربما كانت الوظيفتان الأكثر أهمية في أي مشروع برمجي هما مدير المنتج (أو مدير المشروع للمشاريع الكبيرة) ومدير التصميم.

وقد تمَّتْ مناقشة الأدوار والعلاقات بين مدير المنتج ورئيس التصميم في (Hofmeister [et al.], 2000 and Paulish 2002) على التوالي. ويهتم رئيس التصميم بشكل أساسي بالأمور التقنية المتنوعة للمشروع، بينما يهتم رئيس المنتج بالأمور الإدارية المتعلقة بالمشروع كالإطار الزمني والموازنة والتنظيم وتشكيل فرق العمل ووضع العمل. في المشاريع البرمجية الموزَّعة، يجب أن يتخذ مدير المنتج ورئيس التصميم قرارات حول الفريق الموزع في عدة مناطق جغرافية والذي لا يعمل تحت إشرافهم المباشر. لذلك، وبالإضافة إلى المهارات الإدارية والتقنية الأساسية المعتادة، يجب أن يكون كل من مدير المنتج ورئيس التصميم قادرين على العمل مع أفراد فريق عمل مكوّن من عدة شركات ومن ثقافات متنوعة. ستتطور مهارات التواصل لديهم بما إنهم يقومون بقيادة فريق عمل لم يقابلوا أفراده من قبل ولديه دافع أدائي يختلف عن دافعهم. ونوصي بأن يكون لدى الفريق المكلف بهذه الوظائف خبرات ثقافية أو أن يُعطَى دورات في هذا المجال في مرحلة مبكرة من المشروع. إضافة إلى ذلك، يجب أن يتمتعوا بالمرونة والقدرة على التكيف كلما تقدم المشروع وكان هناك تغيرات على الوضع العام خارجة عن إطار السيطرة، كالظروف السياسية في الدولة التي يوجد فيها فريق البرمجة المسؤولين عنه.

الجدول 9-2: وظائف ومسؤوليات فريق برمجة الوحدات

الوظيفة	المسؤولية
مدير البحث والتطوير	يقوم بوضع خطة دورة حياة الوحدة التي سيتم برمجتها ويعمل على إدارة فرق المشروع.
مصمم الهيكلية	يعمل كعميل مفوّض يعمل على حل الأمور المتعلقة بمجال العمل أو الهيكلية، يتّخذ قرارات تقنية تتعلق بهيكلية الوحدة وتصميمها.
المصمم	يعمل على تصميم الوحدات باستخدام أنماط التصميم المناسبة.

المبرمج	يطبق الوحدات ويتحقق من الشيفرة البرمجة ويعيد كتابتها إن لزم الأمر لضمان جودتها.
خبير ضمان الجودة	يختبر الوحدة كمجموعة واحدة حين برمجتها.

يتم التحكم بالتواصل والتنسيق الكلي للمشروع عن طريق الفريق المركزي. يتم التحكم بالأنشطة التقنية عن طريق قادة الفرق التقنية الوظيفية. يتم التحكم بأنشطة المشروع عن طريق مدير المنتج ومدير المشروع ومديري التزويد. يتم وضع خطة المشروع بحيث يكون عبارة عن فترات برمجية تكرارية ومستمرة مكوّنة من مراحل شهرية. ويتم تحديد أهداف كل مرحلة زمنية مركزياً بالاعتماد على خطة البناء وخطة التكامل والدمج وخطة الاختبار. ونوصي بأن يتم الإبلاغ عن خطط المشروع على مستوى متقدم. ويجب أن ينجز التخطيط بشكل تفصيلي عن طريق كل فريق وحدة في كل مرحلة زمنية. يجب أن تكون مواعيد التسليم محددة، لكن قد يكون هناك نقل لبعض الخصائص من مرحلة زمنية إلى أخرى. هذا وتكون عملية التكامل والدمج من مسؤوليات مطبق عمليات الإنشاء بشكل أساسي وهو عضو في الفريق المركزي ويجب أن يكون على علم وافٍ بمخطط العمل.

فكرة مفيدة:

إحرص على مشاركة الجميع.

من الصعب عادةً قياس مستوى المشاركة وفهم جميع الأعضاء للفريق الذي يعمل عن بُعد. غالباً ما يكون أحد الأشخاص متحدثاً باسم الفريق ما يعني اختصار الطريق أمام أعضاء الفريق الذي يعمل عن بُعد للوصول إلى الهدف عن طريق الفريق المركزي. ومن الوسائل المتبعة

للتغلب على ذلك أن يتم تحديد وظائف لجميع أعضاء الفريق (إضافة إلى كونهم مبرمجين)، ويطلب منهم إعداد تقارير عن بعض حالات العمل، والإجابة عن أسئلة تتعلق بالوظائف الموكلة لهم. قد يساعد هذا في ضمان مشاركة جميع أعضاء الفريق وإتاحة الفرصة لاكتشاف أي أمور قد تطرأ مبكراً.

9-2 الحجم

اقترحنا في الشرح السابق أن لا يزيد عدد أفراد الفريق على عشرة أشخاص، وأن لا يكون في الموقع الواحد أكثر من عشر فرق. وقد استُنتجت هذه القاعدة عن طريق التجربة. تعرفنا على حالات تعدى فيها عدد الأشخاص في الموقع الواحد ثلاثمئة شخص، وعلى الرغم من ذلك استمروا في العمل بنجاح؛ وقد بدأت جميع هذه المشاريع صغيرة. إن اعتماد أسلوب التوزيع الجغرافي في برمجة التطبيقات هو نموذج تغيير أساسي وتحتاج الشركة إلى وقت لتطبيقه. وما لا شك فيه أن الانتقال إلى مشروع برمجة موزع ضخم من دون خبرة سابقة أمر خطير. لذا نوصي مبدئياً أن يكون عدد المواقع في البداية موقعين أو ثلاثة مواقع بحيث لا يزيد عدد المبرمجين على مئة في جميع المواقع. في مرحلة الانتظام في العمل، لم يسبق أن رأينا أي مشروع ناجح، في عدد من المواقع، يزيد على سبعة أو ثمانية مواقع.

إذا كان لديك مشروع يتعدى حجمه الحدود القصوى المقترحة التي تم ذكرها سابقاً، ننصحك باتباع أسلوب المناوبة في العمل لخفض العدد الكلي للفريق. ومن الآراء المقترحة زيادة البرنامج الزمني للبرمجة وتضمين مناوبات إضافية للأعضاء. كما ينصح رأي

آخر بأن يتم تقسيم النظام إلى أنظمة جزئية مختلفة بحيث تصبح مشاريع متعددة ومستقلة. وقد ساعد اعتماد أسلوب خطوط الإنتاج الكثير من المشاريع في تطبيق هذه الطريقة من التقسيم بشكل فعلي.

فكرة مفيدة:

تأكد من أنه يمكن استيعاب كل فريق في غرفة واحدة. ثمة رأي ناتج من تجربة يبين أنه يجب ألا يزيد عدد أفراد فريق مبرمجي الوحدات على عشرة مبرمجين؛ لو كان هذا الفريق يعقد اجتماعات يومية بشكل غير منتظم، سيكون هناك حاجة ملحة لمكان لعقد الاجتماعات. لذا، سوف يحتاج المشروع إلى قاعة مؤتمرات أو إلى غرف عمل خاصة لعقد الاجتماعات، وبالتالي سيكون حجم الفريق محدوداً بحجم الأماكن المخصصة المتوفّرة. إذا لاحظت أن الفريق يجتمع في مكان واحد غير أن التواصل بين أعضائه غير منظّم وغير مريح، يجب أن تعيد تنظيم الأمور في هذه الحالة.

9-3 الملخص والاستنتاجات

سوف يكون تنظيم مشروع التطبيقات الموزَّعة الخاص بك مختلفاً عن تنظيم المشروع الموجود بشكل كامل في مبنى واحد حيث تعمل. على الرغم من أنك ستتبع العديد من الممارسات الإدارية التي كانت مفيدة وتعمل بشكلٍ جيدٍ في السابق، فإنه يجب أن تعمل على التعويض عن القدرة على التواصل الفعّال مع فرق البرمجة التي تعمل عن بُعد. إضافة إلى ذلك، سيصبح لديك قدرة أقل على السيطرة على المشروع وسوف تعمل مع فريق عمل لم تتعرف عليه وقد لا تقابله نهائياً. ستستعين بمديري التزويد لديك

كأعضاء فعَّالين لتقليص الهوَّة بينك وبين الفرق التي تعمل عن بُعد في ما يتعلق بالتواصل. كما إنك ستحافظ على ديمومة العلاقة بينك وبين المزودين لأمد طويل لأنهم سيصبحون امتداداً أساسياً لشركتك مع مرور الوقت.

9-4 أسئلة للمناقشة

1. ما أهمية التعرف الشخصي على بعض الأعضاء المميزين في الفريق الذي يعمل عن بُعد قبل الشروع في مرحلة إنشاء المشروع؟
2. كيف يمكنك إعداد تنظيم لمشروع موزع فعال بحيث يتعدى حدود الشركة والدولة؟
3. ما هي الأدوار الوظيفية المهمة التي تقوم بأخذ القرارات التقنية والإدارية في المشروع؟
4. في أي مرحلة من مراحل المشروع تكون المجموعات المنتظمة ضرورية؟

المراجع

Books

Hofmeister, Christine, Robert Nord and Dilip Soni. *Applied Software Architecture*. Boston; MA: Addison-Wesley, 2000.

Paulish, Daniel J. *Architecture-Centric Software Project Management*. Boston; MA: Addison-Wesley, 2002.

Conferences

Herbsleb, James D., Daniel J. Paulish and Matthew Bass. «Global Software Development at Siemens: Experience from Nine Projects.» *Proceedings of the International Conference on Software Engineering (ICSE 2005)*, 2005, pp. 524-533.

الفصل العاشر

المدير المزوِّد

وُلدت آن في ولاية وسكنسون والتحقت بجامعة بيدرو، وتعمل الآن في شركة برمجيات كبيرة خارج مدينة شيكاغو. على الرغم من أنها قامت بزيارة العديد من الأماكن في الولايات المتحدة الأميركية وأوروبا أثناء رحلات العمل، وزارت مدينة ديزني الترفيهية مع العائلة في العطلة، غير أنها لا تزال تعتبر نفسها مواطنة منتمية إلى شمال الولايات المتحدة.

تحرص الشركة التي تعمل فيها آن على خفض تكاليف المنتجات البرمجية عن طريق نقل بعض أعمالها إلى شركات خارجية في الهند ودول أوروبا الشرقية. وكنتيجة لعمل آن الناجح في إدارة مجموعة برمجة صغيرة، فكَّر مديرها في منحها وظيفة جديدة كمديرة تزويد. لم تفهم آن ما هي طبيعة وظيفة مدير التزويد، لكن مديرها حذَّرها من أن وظيفة مدير التزويد تتطلب السفر بشكل مستمر إلى مواقع الفرق التي تعمل عن بُعد، وأنها يجب أن تشد الرحال وتتحقق من تاريخ انتهاء جواز سفرها حالاً.

نَصف في هذا الفصل الوظيفة المهمة المسؤولة عن إدارة مواقع البرمجة الخارجية. فإن مدير التزويد هو أحد أعضاء الفريق المركزي يقوم بإدارة ثمانية إلى عشرة مبرمجين تقريباً في مواقع البرمجة التي تعمل عن بُعد. وقد تم تطبيق نماذج تنظيمية مختلفة حول المكان الذي يجب أن يوجد فيه مدير التزويد ضمن تنظيم المشروع الموزع. وسوف نناقش الخصائص التي يجب أن تتوفّر في مدير التزويد الفعّال وبعض الصعوبات التي قد تواجه من يعمل في هذه الوظيفة.

10-1 الوظائف والمسؤوليات

لوصف وظائف ومسؤوليات مدير التزويد، يجب في البداية أن نوضح بعض التعريفات البرمجية التي تم اعتمادها وكما وصفها كوكبيرن (Cockburn, 2002):

- **البرمجة في مواقع مختلفة**: يتم تنفيذها في عدد قليل نسبياً من المواقع مع وجود مجموعات كاملة من المبرمجين الذين يقومون ببرمجة الأنظمة الفرعية بوجود واجهة معرفة جيدة للنظام الفرعي.
- **البرمجة عن بُعد**: يرسل المصممون المواصفات والاختبارات من موقع واحد إلى المبرمجين في دولة أخرى. وتفتقر المواقع البعيدة إلى المخططين والخبرات الفنية والمصممين ومختبري البرامج.
- **البرمجة الموزَّعة**: لقبت مجازاً «البرمجة التي لا تغيب عنها الشمس». بما إن أعضاء الفريق الواحد يمكن أن يوجدوا في أماكن كثيرة بحيث تستمر البرمجة على مدار الساعة، يتطلب الأمر تعاوناً بين جميع أعضاء الفريق لتحقيق التواصل اليومي. البرمجة ذات المصادر المفتوحة هي نوع من البرمجة الموزَّعة مع وجود فروق فلسفية واقتصادية ونموذج بنية الفريق. ويمكن لأي شخص في هذا النوع من البرمجة المساهمة بكتابة الشيفرة البرمجية، غير أن هناك حماية لقاعدة البرمجة الأساسية.

في منهجنا من أجل تطوير البرمجيات الموزَّعة المراكز، أرسلنا المواصفات إلى المواقع التي تعمل عن بُعد في البلاد الأخرى، لكن في نفس الوقت قمنا ببناء عناصر الخبرة في الفرق التي تعمل عن بُعد عن طريق مشاركتهم في أنشطة المراحل الأولية في الموقع المركزي. يساعد مدير التزويد في بناء هذه الخبرات عن طريق تكوين العلاقة بين الفريق المركزي والفريق الذي يعمل عن بُعد.

في شركة سيمنز، نعرّف البرمجة بالاستعانة بالمصادر الخارجية (outsourcing programming) على أنها القيام ببرمجة أحد أجزاء المنتج البرمجي خارج نطاق مبنى شركة البرمجة المحلية أو في أحد أقسام شركة سيمنز الأخرى. ويشير مصطلح البرمجة الاستعانة بالمصادر الخارجية عادةً إلى البرمجيات التي يتم تنفيذها خارج مبنى الشركة. على كل حال، تمتلك شركة سيمنز فروعاً كاملة أو جزئية في دول كالهند والبرازيل والصين وأوروبا الشرقية حيث تنفّذ البرمجة بالاستعانة بالمصادر الخارجية.

تتم البرمجة والتواصل من خلال شبكة داخلية مشتركة، ويتم الإفصاح عن الإنجازات بحيث تُكتب بطريقة مشتركة بين أعضاء الفريق والشركة الفكرية المملوكة من قبل الشركة الأم. ويتم إنجاز العمل عن طريق عقد داخلي بسيط يتم من خلاله تحديد الأهداف والمنجزات والتكاليف وطرق الدفع وسياسات تدرُّج العقوبات وإجراءات فصل الموظفين عند الحاجة. ويتم الاتفاق على أسلوب الحوافز والخصومات لمكافأة أو معاقبة فريق يعمل عن بُعد على أساس أداء الفريق. ولكن يتم صياغة معظم العقود بحيث تكون مكافآت الشركة المزوِّدة للعاملين عن بُعد جيدة لتحافظ وترفع التمويل من التنظيم المركزي بمرور الوقت. أما الوضع الأفضل فهو أن يتم تمويل الفريق الذي يعمل عن بُعد، صاحب الأداء الجيد بشكل متكرِّر خلال سنوات مالية متعددة، ويتم استدعاء الفريق

للمشاركة في صيانة المنتج الذي قاموا ببرمجته والمشاركة في أعمال برمجية جديدة. لذلك، يقوم مدير التزويد بتسهيل نقل التقنية من الموقع المركزي إلى المواقع التي تعمل عن بُعد بحيث تصبح هذه المواقع منافِسَةً للموقع المركزي بمرور الوقت. ويعمل هذا النموذج بشكل فعّال في شركة سيمنز بافتراض أن العمل التجاري لديهم ينمو باستمرار. وأما إذا كان العمل التجاري متراجعاً، يتم التخلص من المواقع التي تعمل عن بُعد كإجراء أولي.

تُعرّف عملية تطوير البرمجيات الموزَّعة المراكز على أنها مشاركة فرق البرمجة الموزَّعة في عدة مواقع. وتكون هذه المشاركة من فرق من شركات مختلفة ودول مختلفة. لذا، يجب أن يتمتع مدير التزويد بمهارات القدرة على العمل والتواصل بفاعلية مع مهندسين ومبرمجين ذوي ثقافات مختلفة ومن دول تختلف عن دولته. من الواضح أنه يجب أن تتوفّر لدى مدير التزويد خاصيَّة أخرى، وهي الرغبة في السفر والتنقل والاستعداد له.

هناك بعض المخاطر التي قد تثير قلقاً حول مشاريع تطوير البرمجيات الموزَّعة المراكز مقارنة بالمشاريع التي تتم في موقع واحد بحيث يساعد مدير التزويد في العمليات الإدارية؛ وتتلخص هذه المخاطر في ما يأتي:

- ضعف القدرة على التحكم بالجدول الزمني للعمل والتكاليف ومستوى أداء العمل.
- احتمال وجود نقص في الكفاءات الأساسية.
- الوقت والمسافة.
- عدم القدرة على فهم الثقافات، والتواصل، واللغة.
- درجة التعقيد.
- سعر صرف العملة.

- السياسة.

هناك ثمة مخاطر سياسية تحيط بالعمل الذي يتم ضمن أطر دولية ليس لها أثر في مشاريع التطوير التي تتم في موقع واحد. فقد يتمكن مدير التزويد من التعامل مع بعض هذه المخاطر، وقد يكون بعضها فوق حدود قدرته. وحين حدوث أي أحداث سياسية غير متوقعة، يؤدي مدير التزويد دوراً مهماً في نقل صورة عن المخاطر المحتملة إلى الفريق المركزي ويعمل على وضع خطط عمل بديلة. ومن الأمثلة على المخاطر السياسية:

- **العمالة:**
  - الدول الصناعية مقابل دول العالم الثالث.
  - الموظفون مقابل الذين يعملون في خطوط الإنتاج.
  - السلع الاستهلاكية مقابل التخصصية.
  - مديرو تقنية المعلومات والمديرون الماليون والمديرون التنفيذيون مقابل المبرمجين.
  - الأعمال التجارية الكبيرة مقابل العمال.
  - الأعمال الآلية مقابل الحرف اليدوية.
- **الحكومة:**
  - أسلحة الدمار الشامل.
  - الهجرة.
  - التعليم.
- **الصناعة:**
  - الملكية الفكرية.
  - ميزان القوى.
  - العلاقة بين المنتجين والمزوِّدين والبائعين.

أصبح التنسيق والتحكم أكثر صعوبة بسبب المسافات الكبيرة التي تفصل بين فرق العمل في مشاريع تطوير البرمجيات الموزَّعة المراكز جغرافياً. وتتلخص وظيفة مدير التزويد في تقليص هذه المسافات عن طريق العمل كرابط بين التنظيم المركزي والتنظيم الذي يعمل عن بُعد. ويساعد العمل الذي يقوم الفريق المركزي بطلبه من فريق البرمجة الذي يعمل عن بُعد في تخطيط وتقدير الجهد اللازم لإنجاز الوحدات التي يقومون ببرمجتها إلى حد ما. على كل حال، سوف يساعد مدير التزويد ـ وهو موظف لدى الموقع المركزي ـ على إدارة العلاقات والمشاريع مع الفريق المركزي.

إن التواصل الشخصي مع مدير التزويد أمر ضروري لدعم التواصل مع الفريق الذي يعمل عن بُعد خلال عملية البرمجة. ومن الواضح أن المعرفة السابقة الشخصية تكون ضروريِّةً عند حدوث أي عوائق في تنفيذ المشروع. إن إيجاد الحلول لمشاكل المواقع التي تعمل عن بُعد أمر صعب إذا لم يكن العاملون في الموقع المركزي والعاملون في الموقع الذي يعمل عن بُعد قد التقوا أو عملوا مع بعضهم من قبل.

يمكن تلخيص مسؤوليات وظيفة مدير التزويد في مشاريع تطوير البرمجيات المختلفة كالآتي:

إدارة الموارد (تقريباً من ثمانية إلى عشرة أفراد) والمشاريع في مواقع تطوير البرمجيات التي تعمل عن بُعد، وإدارة العلاقة بين المزود والتنظيم المركزي، وتسهيل التواصل بين الفريق المركزي والفريق الذي يعمل عن بُعد.

10-2 المهارات المطلوبة

يمتلك مدير التزويد العديد من مهارات مدير المشروع البرمجي ومهارات أخرى أيضاً. قمنا بنقل مهام ومهارات مدير مشروع

البرمجيات في الجدول 10−1 من المصدر (Paulish, 2002).

سيكون مدير التزويد بحاجة إلى العمل مع الأعضاء من كلا الفريقين المركزي والذي يعمل عن بُعد. لذلك، يجب أن تكون مهارات الاتصال والتواصل الثقافي التي يمتلكها أكثر كفاءة منها لدى مدير مشروع يقوم بإدارة موقع واحد فقط. ومن الأمثلة على المهارات التي يمكن استخدامها بشكل فعَّال من قبل مدير التزويد مهارة التحدث بعدة لغات. ويمكن أن يتم إنجاز العمل باللغة الإنجليزية، لكن يمكن أن يستخدم مدير التزويد لغة الفريق الذي يعمل عن بُعد للتواصل الاجتماعي بشكلٍ أفضل. إضافة إلى ذلك، من المفيد فهم ثقافة الدولة التي يعمل فيها الفريق الذي يعمل عن بُعد.

كما إن امتلاك مدير التزويد درجة عالية من المرونة والقدرة على الحركة أمر مفيد للغاية. وعلى الأرجح سيقوم مدير التزويد برحلات طويلة ومتكررة إلى موقع الفريق الذي يعمل عن بُعد. ومن الأفضل أن يشعر مدير التزويد بأن مثل هذه الرحلات ممتعة، وليست مجرد روتين مُملٍّ. نشجّع مدير التزويد والأعضاء الآخرين الذين يعملون لدينا في مشروع التطوير الموزع للمشاركة في رحلات جوية وبرامج فندقية دائماً. إذ إنه لا تتم رحلات السفر في بعض الأحيان كما هو مخطَّط لها، فلذا تساعد المرونة في تجنب المشقة المحتملة.

في مشروع نظام إدارة المباني الآلي، لوحظ أنه عند ترتيب اجتماعات في دولة أجنبية حول المشروع، يتأخر بعض الأعضاء بسبب مواعيد رحلات الطيران؛ ما يجعل الاجتماع غير فعّال ويعكِّر صفو الأجواء والأمزجة، ويُؤَدِّي إلى نفاد الصبر وغير ذلك من الآثار السلبية. ولهذا السبب، ننصح بتنظيم وظائف الفريق المركزي بحيث يكون مدير التزويد قادراً على التغلب على الصعوبات المتعلقة بالسفر والتنقل.

الجدول 10-1: وظائف مدير المشروع

المهمة	الوصف	المهارات المطلوبة
تحديد رؤية المشروع	يجب أن يكون مدير المشروع قادراً على عرض رؤية ناجحة لعملية إكمال المشروع.	التسويق، بيئة العمل التجاري، اتجاهات تقنية، خبرات من مشاريع سابقة، التواصل، تحديد الأهداف، تكوين إجماع على الآراء.
موجّه، ومدرب، ومراقب	يجب أن يمتلك مدير المشروع مهارة التوجيه، ويعمل كمراقب لأعضاء الفرق التي لا تمتلك الخبرة، أو كمدرب لأعضاء الفرق التي تمتلك خبرة أكبر، ويقوم بالتطوير تبعاً لظروف العمل.	قادر على التواصل، متعاطف، يمتلك مهارات تقنية
صنع القرارات	يجب أن يأخذ مدير المشروع قرارات في أوقات معينة حول المشروع بحيث لا يعمل على وقف استمرارية عمل أعضاء الفريق.	التحليل الشامل، التواصل، تحليل المبادلات، بيئة العمل التجاري، البيئة السياسية.
التنسيق	يجب أن يقوم مدير المشروع بتنسيق التواصل الضروري بين أعضاء الفريق، ويوفر الدعم التنظيمي للحيلولة دون إضاعة وقت أعضاء الفريق.	إدارة الوقت، التنظيم، التواصل .
العمل مع أعضاء الفريق	يجب أن يتفاعل مدير المشروع مع جميع أعضاء الفريق وبخاصة رئيس التخطيط.	إنشاء الفريق، عضو في الفريق، التواصل، تكوين إجماع على الآراء، الشخصية الذاتية، اجتماعي، مثقف.

المصدر: بيرسون للتعليم (Pearson Education).

سوف تعود الفائدة على مدير التزويد نتيجة اكتسابه مهارات جيدة في التواصل الاجتماعي. من المعتاد في شركة سيمنز أن يتم دعوة الزائرين للمواقع التي تعمل عن بُعد إلى عشاء أو مناسبات اجتماعية خلال فترة زيارتهم. نحن نشجع مثل هذه العلاقات الاجتماعية لأنها تساعد في بناء عمل جماعي أفضل.

قد يحتاج مدير التزويد مهارات تقنية كثيفة أكثر من مدير مشروع يعمل في موقع واحد. إذا احتاج مدير المشروع الذي يعمل في موقع واحد أي معلومات تقنية، يمكنه بكل بساطة التوجه إلى أحد الخبراء وسؤاله عن ذلك. وقد يواجه مدير التزويد للمواقع التي تعمل عن بُعد مشكلة في فارق التوقيت بين موقعه والموقع المركزي، إذ قد يكون أعضاء الفريق المركزي نياماً عند الحاجة إلى معلومات تقنية منهم. لذا، لا بد أن يمتلك مدير التزويد مهارات تقنية جيدة وخبرة جيدة أيضاً في هذا المجال. في الواقع، لقد قام الكثير منهم بالمشاركة في المراحل المبكرة في نتائج البرمجة التي سيتم إعادة تسليمها إلى فريق البرمجة الذي يعمل عن بُعد، وبالتالي سيكونون مدركين لمتطلبات المنتج وهيكلية النظام.

فكرة مفيدة:

راقب كيفية تقديم الأشخاص أنفسهم في اجتماع الموقع الذي يعمل عن بُعد.

سوف تتعلم أكثر عن المزودين إذا قام أعضاء الفريق بتقديم أنفسهم إلى بعضهم بعضاً وإليك أيضاً، إذ لا بد أنهم لم يتقابلوا مسبقاً. وفي بعض المواقع، لا يساعد تغيير أعضاء الفريق السريع في تكوين علاقات عمل طويلة الأمد بين الأعضاء. بما إنك تستثمر في مدير التزويد لمشروعك بحيث يصبح منافساً قوياً لك على

المدى البعيد، قم بكل ما تستطيع لتكوين فرق ذات إنتاجية عالية، واحتفظ بهم للعمل معك لفترة طويلة من السنين.

10-3 نماذج تنظيمية

تشير خبراتنا إلى أن مديري التزويد يجب أن يكونوا مسؤولين عن ثمانية مبرمجين يعملون عن بُعد على الأقل، وأن لا يزيد عددهم على عشرة؛ ومن الأفضل أن يكون مديرو التزويد جزءاً من الموقع المركزي ومقيمين فيه. ولكن يُتوقع أن يقوموا برحلات متكررة إلى الموقع الذي يعمل عن بُعد. يلخص الجدول 10-2 عدداً من البدائل المتاحة لمدير التزويد الذي يعمل في التنظيم وإيجابيات وسلبيات هذه البدائل.

يتم التحكم بالتواصل والتنسيق لمشروع التطوير الموزع عن طريق الفريق المركزي. يتم التحكم بالأنشطة التقنية عن طريق قادة الفرق الفنية الموجودين في الفريق المركزي. أما أنشطة المشروع فيتم التحكم بها بواسطة مدير الإنتاج ومديري التزويد. ويتم التخطيط للمشروع بحيث يتكون من عدد من الفترات البرمجية التكرارية ليتم إنجاز جزء من البرمجية في نهاية كل شهر. ويتم التخطيط للأهداف التي سيتم تحقيقها نهاية كل شهر مركزياً عن طريق خطة بناء المشروع.

يجب أن يقوم كل فريق برمجي بتحقيق المخطط المفصَّل في نهاية كل مرحلة برمجية، وبدعم من مدير التزويد. وتكون المواعيد المتفق عليها للتسليم ثابتةً، لكن سيكون هناك تداخل بين مواعيد التسليم مستقبلاً.

الجدول 10-2: الخيارات البديلة لمدير التزويد

موظف لدى فريق	مقيم في فريق	الإيجابيات	السلبيات
مركزي	مركزي	سيطرة قوية	تواصل أقل مع المهندسين العاملين عن بُعد
مركزي	يعمل عن بُعد	■ معلومات أقل عن المشروع ■ التواصل	■ تواصل أفضل مع المهندسين الذين يعملون عن بُعد ■ تكاليف المعيشة
يعمل عن بُعد	مركزي	■ معلومات أكثر عن المشروع ■ التواصل	تكاليف أكبر يتحملها المزود
يعمل عن بُعد	يعمل عن بُعد	سيطرة أكبر على الفريق الذي يعمل عن بُعد	■ معلومات أقل عن المشروع ■ التواصل

اعتماداً على ما سبق، نعتقد أن البديل الأفضل هو أن يكون مدير التزويد جزءاً من الفريق المركزي وموجوداً فيه. وسوف يكون هناك رحلات متكررة إلى المواقع التي تعمل عن بُعد، خصوصاً في الاجتماع المتعلق ببداية دورة جديدة وفي عرض نتائج نهاية فترة العمل. لذلك، يجب أن يتم التخطيط لمدير التزويد للقيام برحلة واحدة إلى المواقع التي تعمل عن بُعد في كل شهر، أو بمعنى آخر، يُخصَّصُ ما نسبته 25 بالمئة من وقت عمل مدير التزويد للمواقع التي تعمل عن بُعد. إضافة إلى ذلك، قد يقوم الخبراء التقنيون في الموقع المركزي بزيارات لاستعراض ومتابعة سير العمل وعقد ورش عمل أو مراجعة وتدقيق إنجاز الفريق الذي يعمل عن بُعد. إن قيام الفريق الذي يعمل عن بُعد بعرض شهري للعمل من شأنه مساعدة مدير

التزويد على تقدير سير أداء العمل للفريق، ويعمل عادة على زيادة ثقة أعضاء الفريق بأنفسهم لأنهم يتطلعون إلى التقدم في العمل على الصعيد الشخصي والتقدم الإجمالي للمشروع.

يفضل المزودون توكيل شخص من فريق العمل لديهم في الموقع المركزي حتى بعد البدء بالبرمجة. إذ يستعينون بهذا الشخص لسد الفجوة في التواصل بين الفريق المركزي والفريق الذي يعمل عن بُعد. على الرغم من أن هذه الوظيفة مفيدة، غير أننا لا نتوقع أن يتمكن هذا الشخص من تغطية الوظائف التي تم تحديدها لعمل مدير التزويد كلياً. قد يمتنع الفريق المركزي عن دفع راتب لهذا الموظف وذلك لوجود عدد كافٍ من الموظفين العاملين كمديري تزويد. ويمكن أن يعمل هذا الموظف كبديل إذا كان مدير التزويد مشغولاً في مشاريع أخرى في مواقع تنظيم مركزية عدة في نفس الدولة، ويستطيع هذا الموظف العمل في مشاريع متعددة. وقد لاحظنا أن مزودينا الداخليين يستخدمون هذا الموظف في الولايات المتحدة بشكل فعّال للمساعدة في توجيه فرق العمل في الموقع المركزي في المرحل المبكرة من المشروع. ولاحظنا أيضاً أنهم في العادة يغيِّرون أماكن وجود أعضاء فرق العمل في الولايات المتحدة برفقة أسرهم. وهذا من شأنه الحد من الشعور بالغربة وتجنب الحاجة إلى القيام برحلات متكررة إلى بيوتهم في المراحل المبكرة للمشروع.

ومن الدراسات التي أجريت في شركة سيمنز، خرجنا باعتقاد أنه من المفيد نقل فرق العمل من الموقع المركزي إلى مواقع التنظيمات البرمجية التي تعمل عن بُعد كمندوبين طويلي الأمد. ومن الممكن تطبيق هذا في شركة سيمنز لأن المزودين هم عبارة عن شركات فرعية. ويكون هؤلاء المندوبون موظفين موثوق بهم في

الشركة المركزية ويتم اختيارهم عادة للمساعدة في إنشاء فرع للشركة في الدولة الأجنبية. وبصفة عامة، تساعد معظم التبادلات التي تتم بين فرق العمل في بناء العلاقة بين الفرق المركزية والفرق التي تعمل عن بُعد، خصوصاً إذا لم تسر عملية التطوير أو التواصل على النحو المطلوب، إذ يتناقص التواصل بين الفرق المركزية والفرق التي تعمل عن بُعد، ويمكن أن يتم إنجاز هذا عن طريق التبادل الفعلي لفرق العمل. يجب أن يتم اختيار فريق العمل الذي سيتم تبادله بحيث يمتلك مهارات التواصل، إذ سيعمل أفراده كحلقة وصل بين الفرق المركزية والفرق التي تعمل عن بُعد.

اعتماداً على خبرتنا، نعتقد أنه من الضروري وجود شخصية تفاعلية تقنية للحفاظ على مشروع التطوير الموزع في مساره. لذلك، وعلى الرغم من أن التوثيق الجيد أمر في غاية الأهمية، غير أننا لا نعتقد أن النموذج المثالي للبرمجة عن بُعد، والذي تم شرحه في 10-1 يمكن أن يكون ناجحاً للمشاريع المركبة في تطوير البرمجيات.

فكرة مفيدة:

قم بإرسال أعضاء فريق العمل الموثوق بهم إلى موقع جديد يعمل عن بُعد.

ناقشنا في هذا الفصل أهمية مدير التزويد الذي يعمل لدى الفريق المركزي الذي يقضي جزءاً من وقته في مواقع البرمجة التي تعمل عن بُعد (على سبيل المثال 25 بالمئة). وقد ناقشنا أيضاً قبل ذلك أهمية عمل الأعضاء المميزين في الفريق الذي يعمل عن بُعد في الفريق المركزي خلال المراحل الأولى من العمل. إضافة إلى ذلك، ضع بعين الاعتبار نقل أعضاء من

الفريق المركزي إلى مواقع مهمة في الفريق الذي يعمل عن بُعد. في بعض الأحيان يكون انخفاض عامل التكلفة السبب المؤدي إلى تعيين أفراد ليسوا من ذوي الخبرة تماماً. بوجود فريق عمل موثوق به في التنظيم الذي يعمل عن بُعد، يمكن أن يقوم أفراده بالمساعدة في إجراء المقابلات والتدريب وبناء تفاهم مشترك والمساهمة في الأنشطة المختلفة التي سوف تساعد في تسريع أداء الفرق التي تعمل عن بُعد في العمل في مشروعك.

10-4 قضايا تعدد الثقافات

لاحظنا في العديد من المشاريع أن المتغيرات الثقافية تكون عاملاً مؤثراً في مشروع التطوير الموزع (Kopper, 1993). وقد تعمل هذه العوامل على تقوية أو إضعاف سير المشروع حسب طريقة إدارتها. في شركة سيمنز، يمكن أن تنبع العوامل الثقافية من عدة دول ومن ثقافات مختلفة للشركات، لأن شركة سيمنز امتلكت العديد من الشركات بمرور الوقت.

قد تؤثِّر العوامل الثقافية في قيم أعضاء الفريق البرمجي بما في ذلك الالتزام بالمواعيد والالتزام بالحد الفاصل بين الوقت المخصص للعمل والوقت الشخصي وحل النزاعات، وغير ذلك. على سبيل المثال، كلما ازداد ضغط العمل بسبب اقتراب موعد تسليم المشروع، قد يتهم أعضاء الفريق الذي يعمل عن بُعد بعضهم بعضاً بالتقصير ويتذمرون بقولهم «هم في إجازة بينما نحن بحاجة لإنجاز العمل».

لقد أكدنا الحاجة إلى مدير التزويد لاستيعاب الأمور الثقافية والقدرة على التواصل مع أعضاء الفريق. وعلى كل حال، تمتد هذه الحاجة إلى جميع أعضاء الفريق. فقد لاحظنا أن المشاريع الناجحة تبني ثقافتها الخاصة التي يتم تبنِّيها من ثقافة الدولة والشركة. ويتم تكرار التواصل بشكل مستمر وبطرق مختلفة للتواصل وذلك للمساعدة في التعليم بأساليب ولغات مختلفة. ويقوم أعضاء فرق العمل من المواقع المتعددة بزيارة بعضهم بعضاً بشكل دوري، يتعرفون على بعضهم خارج نطاق العمل.

وقد لاحظنا أيضاً أن المشاريع الناجحة تستثمر الجهد في عملية تشكيل الفريق وفي التدريب متعدد الثقافات. وهذا من المتطلبات الأساسية لمدير التزويد، لكنه مفيد جداً للفريق كله. وعادةً ما تنعكس عوائد هذا التدريب على النواحي العملية كتحديد وسيلة مشتركة لعقد اجتماعات خلال جميع الوقت اللازم للمشروع.

فكرة مفيدة:

قم بتشجيع المشاركة في الأداء الجيد.

من المفيد أن يكون هناك مقارنات بين أعضاء فريقك والمزودين. اجعل هــذه المقارنات محفّزاً إيجابيـــاً وليس سلبياً عن طريق تشجيع «المشاركة في الأداء الجيد». إذا تبين أن إحدى الفرق تستخدم عملية أو أداة أو تقنية مبتكرة، قم بنشر ذلك بين الفرق الأخرى بحيث يتم التعلم من هذه الخبرة. لا توجه السؤال الآتي: «لماذا لا يستطيع فريقك أن يكون بمستوى الفريق A؟» إذا كنت تقوم بإدارة فريق مصدره أكثر من مزوِّد، يمكن أن يكون بينهم تنافس في العمل، لذا يجب أن ينصبّ اهتمامك على استخدام أفضل الممارسات من أجل

مشروعك، وكن متأكداً أن جميع أعضاء الفريق يملكون دوافع للعمل ومنتجون ذوو مستوى متقدم.

10-5 الملخص والاستنتاجات

بصفتك مدير تزويد، أنت عضو مهم في فريق المشروع للمساعدة في سد الفجوة في عملية التواصل مع الفريق الذي يعمل عن بُعد. وسوف تكون نظرتك للعلاقة مع مزوديك على أنها علاقة طويلة الأمد. ومع مرور الوقت يصبحون امتداداً لشركتك الخاصة. ستنمو قدراتك في التعويض عن بُعد المسافات والثقافات واللغة التي لم تتعرف عليها في المشاريع التي يتم إنجازها في موقع واحد. سوف تكون مسؤولاً عن التقدم في سير المشروع في فريق البرمجة الذي يعمل عن بُعد وفي علاقاتهم في العمل مع الموقع المركزي. باختصار، يتمثل عملك كمدير تزويد بأنك «مدير مشروع برمجي متنقل».

10-6 أسئلة للمناقشة

1. ما هي المهارات المطلوبة من مدير التزويد، وفي أي المجالات يتم استغلال هذه المهارات؟
2. كيف يتم توظيف أعضاء الفريق الذي يعمل عن بُعد، وكيف تتم مقابلتهم وتخصيص الوظائف لهم؟
3. أذكر بعض العوامل الثقافية التي قد تؤثر في تقويم فريق العمل المكوّن من عدة دول تعمل في المشروع.
4. كيف يقوم مدير التزويد بالتواصل مع أعضاء الفريق الذي يعمل عن بُعد؟ وما عدد المرات التي يقوم بها المدير بزيارة الموقع الذي يعمل عن بُعد؟

المراجع

Books

Cockburn, Alistair. *Agile Software Development*. Boston; MA: Addison-Wesley, 2002.

Kopper, Enid. «Swiss and Germans: Similarities and Differences in Work-Related Values, Attitudes, and Behavior.» *International Journal of Intercultural Relations*. New York: Pergamon Press, 1993, pp. 167-184.

Paulish, Daniel J. *Architecture-Centric Software Project Management*. Boston; MA: Addison-Wesley, 2002.



القسم الرابع

المراقبة والتحكم



الفصـــل الحادي عشر

ضمان الجودة

أنيطت مسؤولية ضمان الجودة في مشروع نظام إدارة المباني الآلي إلى رون (Ron) مؤخراً، وتم إعلامه أن معظم عمليات البرمجة ستتم في مدينة بانغالور (Bangalore). كان رون محتاراً حول الطريقة التي يمكن أن يضمن بها جودة البرمجيات التي يبرمجها فريق عمل لم يقابله من قبل، وليس لديه أي تأثير في عملية البرمجة التي يقومون بها.

نبحث في هذا الفصل في الاستراتيجيات التي يمكن اتباعها لمراقبة جودة عملية تطوير البرمجيات الموزَّعة المراكز وجودة المنتج البرمجي. ويتم تحسين هذه العملية عن طريق قياسات ومراجعات مستمرة يُحدَّد من خلالها الزمان والمكان اللذان تمت خلالهما عملية البرمجة الخاطئة للعملية؛ ومن الاستراتيجيات المتبعة لتحسين جودة المنتج القيام بتحليل بيانات المقاييس الأساسية ومنع حدوث العيوب والمشاكل في النظام والاختبار المستمر.

11-1 تمهيد

تعرّف هيئة مهندسي الكهرباء والإلكترونيات (IEEE) جودة البرمجيات على أنها:

1. الدرجة التي يتفق معها النظام، والمكوّنات، والعمليات مع المتطلبات المحددة.
2. الدرجة التي يتفق معها النظام، والمكوّنات، والعمليات مع الحاجات المتوقعة من العميل.

تعريف ضمان الجودة (QA) الذي ينسجم مع هذه التعريفات هو:

1. جميع الأنماط المنتظمة والمخطط لها للعمليات الضرورية لتوفير ثقة كافية بأن المنتج أو العنصر يلبي المتطلبات التقنية للمؤسسة.
2. مجموعة الأنشطة المصممة لتقويم العمليات المستخدمة في تطوير أو إنتاج المشروع .

غير أننا نجد أن تعريف هيئة مهندسي الكهرباء والإلكترونيات لجودة البرمجية جيدة، فإننا نرى بأن تعريف ضمان الجودة يحدد هدف الضمان بأن يتم تحقيق المتطلبات التقنية للمؤسسة. وفي رأينا، تشمل أنشطة ضمان الجودة مقاييس لضمان أن متطلبات المؤسسة تعكس بدقة احتياجات العميل أو المستخدم. ونحن نؤمن بما تحمله معتقدات منهجية البرمجة السريعة (Agile) أن المستخدمين والعملاء ليسوا قادرين دائماً على الوصف الدقيق لاحتياجاتهم؛ المقولة القديمة تقول «لا أستطيع إخبارك بما أريده، لكني سوف أعرف ما أريد بمجرد أن أراه». نعيد بالتالي تعريف هيئة مهندسي الكهرباء والإلكترونيات لضمان الجودة ليصبح:

«هو جميع الأنماط المنتظمة والمخطط لها للعمليات الضرورية لتوفير ثقة كافية بأن المنتج أو العنصر يلبي المتطلبات التقنية والاحتياجات المتوقعة من العميل أو المستخدم».

تتضمن عملية ضمان الجودة الإجابة عن الأسئلة الآتية:

- كيف يمكننا أن نتأكد أننا أنتجنا ما نحتاجه ونريده بدقة؟
- كيف يمكننا أن نتأكد أننا أنتجنا نظاماً يحقق المتطلبات المتوقعة؟
- كيف يمكننا أن نتأكد أننا أنتجنا نظاماً يعمل كما هو متوقع منه في الوقت الحالي ومستقبلاً؟
- كيف يمكننا أن نتأكد أننا أنتجنا نظاماً لن يقوم بعمل أي أمور غير متوقعة منه؟

نصف في هذا الفصل الأمور المتعلقة بضمان الجودة في مشاريع تطوير البرمجيات الموزَّعة المراكز وكيفية التعامل مع هذه الأمور ومناقشة صيانة البرمجيات التي أنشئت من خلال تطوير البرمجيات الموزَّعة المراكز. ولن نناقش تفاصيل ضمان الجودة المتعلقة بتطوير البرمجيات الموزَّعة المراكز والتي لا تختلف عن تفاصيل الجودة في المشاريع المنتظمة أو التفاصيل التي ليس لها تأثير في الأنشطة التي تم وصفها في هذا الكتاب.

11-1-1 ضمان الجودة في سياق شامل

لماذا تختلف عملية ضمان الجودة في سياق شامل؟ لماذا لا يكون ذلك فاعلاً إذا وضعنا منهجاً صارماً في عملية ضمان الجودة للمشروع المستهدف؟ فقد تكون عملية ضمان الجودة في تطوير البرمجيات الموزَّعة المراكز مختلفة لنفس الأسباب التي تم ذكرها مرات عديدة في هذا الكتاب. في المشاريع المنتظمة، هناك شبكات أمان غير رسمية تعدّل القصور في العمليات الجارية. وهذا يعني أن

عملية ضمان الجودة ليست العملية الأساس التي تعيد الأمور لنصابها، بل هي عمليات التفاعل المتقاربة والثابتة من قبل أعضاء الفريق للقضايا التي تظهر على السطح (أو ربما كان ذلك ضمن جزء من عملية ضمان الجودة). إن شبكات الأمان هذه غير موجودة في عملية تطوير البرمجيات الموزَّعة المراكز. إضافة إلى ذلك، قد يكون أثر هذه الأمور أكبر في عملية تطوير البرمجيات الموزَّعة المراكز. يشبه ذلك سيارة نقل صغيرة يرتبط بها مجموعة من العربات تسير على الطريق السريع، وعندما يقوم السائق بتعديل الاتجاه، يصبح التعديل أكبر وأكبر حين انتقاله من عربة إلى أخرى. وقد تؤدي التعديلات البسيطة في المسار التي يقوم بها السائق إلى نتائج كارثية على العربات. وينطبق هذا على عملية تطوير البرمجيات الموزَّعة المراكز.

عند نقل مواصفات المشروع إلى الفريق الذي يعمل عن بُعد، لا يمكن التأكد ممّا إذا كان أعضاء الفريق قد أدركوا طبيعة المهام المنوطة بهم. في المشاريع المنتظمة، يتقابل أعضاء الفريق وجهاً لوجه، كما يتم تأسيس علاقات في العمل تجعل التواصل أكثر سهولة؛ عندئذ ستتمكن من الانتباه إلى الملاحظات الضمنية التي تشير إلى وجود ارتباكات غير مريحة في المهام. وفي عملية تطوير البرمجيات الموزَّعة المراكز يتم إرسال المواصفات لاسلكياً من خلال الشبكة الإلكترونية، أو عن طريق شبكة مرئية خاصة، وستكون على يقين أن الفرق ستخبرك إذا كان هناك مشاكل أو استفسارات تتعلق بالمواصفات المستلمة. مقارنةً بالمشاريع الموزَّعة، توفّر المشاريع المنتظمة فرصاً متزايدة لامتلاك رؤية لتنفيذ المهام الفردية. عليك أن تتعرف على الأعضاء المميزين في الفريق وأسلوب العمل الذي يتبعونه ومدى سير الأمور بشكل جيّد. ولا تساعد هذه المعلومات في تحديد القضايا الأساس وحسب، بل في إيجاد حلول لها بشكل سريع.

عند اكتشاف أي مشاكل في مشروع تطوير البرمجيات الموزَّعة المراكز، من الصعب معالجتها كما في المشاريع المنتظمة. وإن من الأمور الأكثر صعوبة تحديد الشخص الواجب إناطة حل المشكلة له، ومن الصعب ضبط الجدول الزمني وتحويل المهام إلى المحاسب لتقدير أي مشاكل في المهام أو جهد جديد غير متوقَّع، وبشكل عام لا تجعل أي مشاكل في المهام تؤثر في سير العمل. اعتماداً على نوع المهمة (على سبيل المثال عدم فهم للمتطلبات، أمور نحوية، أو أمور تتعلق بالسلوك)، سوف يتم اكتشافها في أوقات مختلفة. كلما كنت بعيداً عن المشكلة عن ظهورها، كان صعباً عليك تصحيح مصدر المشكلة. وليس من غير الشائع توكيل مهمة حل مشكلة ما إلى أكثر من شخص لحلها قبل أن يتم حلها بالفعل. وقد صاحب إعادة التكليف بحل المشكلة تحقيق بها، وكذلك تحقيق بأي شخص يكون قد تسبب بالمشكلة. هذا وقد يحتاج ذلك وقتاً إضافياً على حساب المهام الأخرى. وفي نهاية المطاف قد يحتاج الأمر إلى مفاوضات أو تعاون من أعضاء الفريق الذين يصعب أن يؤدوا عملهم بشكل فاعل. وقد يؤدي إعادة تكليف المهام إلى تأثير غير جيد في سير المشروع. وإذا كانت إحدى الفرق تعمل على تطوير أحد الأمور في مرحلة حرجة لتأثيرها في المخطط المستقبلي للمشروع، فإن التأخر في تسليم هذا الجزء قد يعني وجود فرق أخرى بلا عمل في انتظار التسليم من فريق آخر.

11-2 قياس جودة العمليات

أصبح التنسيق والتحكم أكثر صعوبة بسبب التوزيع الجغرافي للفرق. يجب أن نجعل عملية أداء العمليات تؤدَّى بطريقة تضمن الجودة العالية للمنتج. ومن الصعب التأكد من أن ممارسات محددة سوف تكون فعَّالة في مشروع معيّن بسبب الاختلاف الجذري

لمشاريع تطوير البرمجيات الموزَّعة المراكز عن بعضها بعضاً. في مشاريع تطوير البرمجيات الموزَّعة المراكز في شركة سيمنز، لم نجد أي علاقة بين ممارسات معينة تقوم بها الشركة وبين نجاح المشروع. ذلك يعني أننا بحاجة ماسة إلى تعريف محددات تمنحنا مؤشرات تحذيرية مبكرة عند الحاجة إلى إجراء أي تعديلات على أي عملية من عمليات المشروع. ونناقش في هذا الفصل بعض الممارسات التي تم استخدامها بنجاح والمحددات التي قد تكون مفيدة وكيفية تقويمها، ونتيجة لذلك يتم تحسين العمليات على المشروع.

11-2-1 تحديد العمليات

إن من الأمور الحسنة أن تمتلك شركتك مجموعة من العمليات لهندسة البرمجيات (SEPG)، وأن يكون لديها مجموعة نموذجية ومحددة من العمليات من أجل تطوير برمجية معينة؛ قد تحتاج الكثير من التفاصيل للتعديل، وقد يوجد بعض التفاصيل التي لم يتم معالجتها نهائياً، أو قد لا تناسب الفريق الذي يقوم بعملية تطوير البرمجيات الموزَّعة المراكز في المشروع. على سبيل المثال، تصف العمليات التي تُجرى على هندسة المتطلبات الطريقة التي يتم بها صياغة المتطلبات بحيث تسهل العملية الآلية للاختبار. ويجب تعريف جوانب محددة بشكل مختلف من مشروع لآخر وفقاً لطبيعة المشروع، كالبنية التحتية المستخدمة للمشروع والممارسات الإدارية التفصيلية والمستوى التدريبي التطويري اللازم للفريق وسرعة إنشاء المشروع والعمليات اللازمة له، وغير ذلك. ويتم اختيار هذه العمليات بحيث تراعي الموازنة بين المخاطر والنفقات. وكما تمت مناقشته في الفصل السادس (تحليل المخاطر)، يجب أن يتم الاختيار أو إعادة تكييف العمليات بمعرفة تامة بخصائص المشروع (سُمِّي ذلك بـ «ملف المخاطر» في الفصل السادس من هذا الكتاب). وفي

حقيقة الأمر، فإن تعريف هذه العمليات ما هو إلا بداية. فالاختيار هو عملية تكهنية تعتمد دقتها على الكثير من العوامل. ومن الحكمة أن تفكر في المدى البعيد وتضع خططاً للطوارئ بالإضافة إلى عملية الاختيار. فما مدى القدرة على إجراء تعديلات منطقية وقابلة للتطبيق على العمليات إذا كان المشروع لا يسير كما هو متوقع (لمزيد من التفاصيل، انظر الفصل السادس من هذا الكتاب)؟

11-2-2 تحديد المعايير

بعد أن يتم تحديد العمليات، يجب أن تحدد الآلية التي ستستخدم لمراقبة هذه العمليات. ومن الأمور التي تم الاستشهاد بها مرات عديدة صعوبة الحصول على صورة واضحة لعملية التقدم في سير العمل والوضع الحالي لمشروع تطوير البرمجيات الموزَّعة المراكز. ولا تظهر المشاكل عادة إلا في المراحل النهائية للعمل، وهذا يؤدي إلى اختلال كبير في العمل (تخيل المثال السابق للشاحنة الصغيرة). ونحتاج إلى وضع معايير نتمكن من خلالها الحصول على مؤشرات تحذير مبكرة بأن العمل لا يسير كما هو مخطَّط له، ما يساعدنا في التصرف بسرعة قبل ظهور نتائج كارثية. ولا يمكن الحصول على هذا بسهولة. ويجب أن نكون قادرين على وضع دليل قوي ومتماسك لذلك (أبعد من الأهداف الأساس والأسئلة والأسلوب المعياري)، لكن للأسف كل ما نستطيع فعله هنا هو الاستفادة من بعض الاستنتاجات من تجارب سابقة. ومن الأمور التي يمكن مراقبتها وتعتبر من المؤشرات التحذيرية لأمر ما:

- **زيادة التواصل**: عندما يجد الناس مشاكل في عملهم، يبدأون في البحث عن أحد ما لمساعدتهم، وهذا يعني أنهم سيستخدمون الهاتف أو يرسلون رسائل عبر البريد الإلكتروني أكثر من الطبيعي، أو يعقدون اجتماعات مباشرة مع بعضهم

بعضاً. فكيف يمكن أن يساعدنا ذلك؟ في الحقيقة إذا كنت قادراً على ملاحظة أي من هذه الأمور ـ كارتفاع مفاجئ في عدد الرحلات غير المخطط لها من قبل الأعضاء، أو زيادة في عدد مستخدمي مزودي الخدمة، وغير ذلك ـ ربما يستحق ذلك البحث في سبب حدوثه. ويمكن تفسير بعض هذه الأمور عن طريق الجدول الزمني (على سبيل المثال، قد يزداد التواصل خلال الأوقات التي يكون فيها تداخل في نهاية مرحلة سابقة وبداية مرحلة جديدة أو قبل مواعيد التسليم)، لكنها قد تكون مؤشرات على حدوث أمور غير طبيعية.

- **انخفاض غير طبيعي في عملية التسليم عند نهاية إحدى المراحل**: بسبب الطبيعة السلسة لعملية تخطيط العمليات، فمن الطبيعي أن تكون مواعيد تسليم المراحل قابلة للتعديل بشكل ما، لكن يتم ضبط ذلك تماماً مع مرور الوقت. وإذا كان تسليم العمل من إحدى الفرق أقل بكثير مما هو متوقع، فمن الحكمة البحث في سبب ذلك. فقد تتسبب مجموعة من فرق العمل بهذا التأخير أحياناً، ومن الأفضل أن يتدخل الفريق المركزي مبكراً بحيث يكون مدركاً لأي تغييرات ويقوم بالتعامل معها.

- **الروح المعنوية لفريق البرمجة**: لاحظنا وجود علاقة قوية بين توقع فريق البرمجة لسير العمل والبرنامج الزمني الفعلي للعمل (وعلى العكس من ذلك، هناك علاقة ضعيفة بين توقع الرؤساء لسير العمل والبرنامج الزمني الفعلي للعمل). إذا كان تتبُّع مستوى الإحباط عند فريق البرمجة ممكناً، سيكون ذاك مؤشراً جيداً للوضع الحالي للأمور. وسيعلم الأشخاص الذين يتعاملون مع الأمور التي تحيط بها المشاكل بشكل يومي أن هناك مشكلة عالقة قبل معرفة الإدارة بها. إذ من السهل أن تشعر الفرق بعدم الراحة إذا تُركت من دون عمل، أو إذا

شعرت بوجود عمل مجهد. وهذا مؤشِّر على تخطيط ضعيف أو أداء ضعيف من إحدى الفرق الأخرى أو مشاكل في المتطلبات أو التصميم.

- **تغيير مستمر في المتطلبات أو التصميم**: على الرغم من أن هذا يُكتشف في مرحلة مبكرة في المشروع عند بدء عملية التنفيذ، غير أنه من الحكمة البحث في الأسباب والتأثيرات لهذه التغييرات إذا كان هناك تغيير مستمر في المتطلبات أو التصميم

يجب ملاحظة أن فريق العمل الجديد الذي يعمل عن بُعد يحتاج إلى وقت للوصول إلى أفضل إنتاجية. ومن خبراتنا السابقة في مشاريع التطوير الموزَّعة لشركة سيمنز، قد تحتاج هذه الفرق الجديدة إلى وقت من ستة إلى اثني عشر شهراً لتحقيق أفضل إنتاجية. ويجب أن تكون الأهداف التي يؤمل تحقيقها في أول مرحلة من مراحل العمل متواضعة ويزداد حجمها بمرور الوقت.

11-2-3 تحسين العمليات

بما إنك المدير المركزي لضمان جودة عمل الفريق، تقع على عاتقك مسؤولية التحسين المستمر لعملية التطوير الموزَّعة. ويجب أن يتم أداء هذا بشكل تدريجي وبحذر بحيث لا يتم التشويش على سير عملية البرمجة. وسوف تعمل على هذه التحسينات مع الفريق المسؤول عنها الذي يعمل عن بُعد. كما يتم أيضاً عرض هذه التحسينات مع الإدارة المحلية التي تعمل لديها.

11-3 قياس جودة المنتج

يتم قياس جودة المنتج عن طريق تتبع عدد العيوب من خلال الاختبارات الإضافية للإصدار السابق (Paulish, 2002). يتم الحصول على عدد العيوب باستخدام أداة للتعامل مع التغيرات بحيث تكون

قادراً على تحديد الأولويات وعدد العيوب وحالاتها التي بلّغ عنها فريق الاختبار. ستكون لديك معايير لقياس الجودة مرتبطة بقبول العميل للمنتج؛ على سبيل المثال، «لن يتم تسليم أي منتج إذا كان هناك عيوب ذات مستوى أولوية أولى أو ثانية ما لم تحلّ أو تصحح بعد».

11-3-1 أنواع العيوب

إن مصطلح «عيوب» هو أكثر شمولاً من مصطلح «خطأ»؛ ويعرّف على أنه أي انحراف في المنتج عن «مواءمة الأهداف المطلوبة». وكما تم مناقشته سابقاً، يتم تعريف مواءمة الأهداف المطلوبة بأنه قدرة المنتج على تحقيق التوقعات المطلوبة من المستخدم. لذلك، يمكن تحديد أنواع العيوب كما يأتي:

- **أخطاء برمجية**: أخطاء في لغة البرمجة.
- خصائص غير صحيحة: وهي أخطاء في عملية التصميم المنطقي؛ ويتم الإشارة إليها على أنها أخطاء ذات دلالات لغوية غير صحيحة.
- **خطأ في فهم الخصائص**: يكون ذلك نتيجة فهم خاطئ من المبرمج أو المصمم لمتطلبات العميل.
- **سلوك النظام**: يظهر ذلك عندما يكون أداء النظام ضعيفاً، أو عدم ظهور نهايات متوقعة للبرمجية، أو أن لا يكون أداؤه كما هو متوقع تحت ظروف معينة (على سبيل المثال، عند إجراء إدخال خاطئ).

11-3-2 أمور تتعلق بجودة المنتج في مجال تطوير البرمجيات الموزَّعة المراكز

إذا أعدنا النظر في العيوب السابقة، سنرى بوضوح أموراً أخرى

تحافظ على جودة المنتج في ضبط عملية تطوير البرمجيات الموزَّعة المراكز. ويحدث ذلك عندما يُحدِّد مترجم لغة البرمجة وجود بعض الأخطاء في لغة البرمجة (مثلاً، عدم وجود توافق في عملية ربط المتغيرات)، ذلك يعني أنه من السهل تحديد هذه الأخطاء حتى في عملية تطوير البرمجيات الموزَّعة المراكز، لكن أخطاء الدلالات اللغوية هي أكثر صعوبة بكثير. يجب أن يكون لدينا أساليب مناسبة للإحاطة بحالات الاختبار التي نجريها للكشف عن أخطاء الدلالات اللغوية في عملية تطوير البرمجيات الموزَّعة المراكز وتصحيحها. ففي حين ينطبق هذا أيضاً في المشاريع المنتظمة، غير أن احتمالية مواجهة مثل هذه المشاكل يكون أكبر، واحتمال أن تؤدي تأثيراتها إلى حدوث عيوب أكبر بكثير في عملية تطوير البرمجيات الموزَّعة المراكز. فقد تؤدي المفاهيم المختلفة بين الفرق مثل الدولار الأميركي مقابل اليورو (كما حدث في المركبة الفضائية التي كان يفترض أن تهبط على المريخ) إلى أخطاء منطقية ومشاكل محصورة في فريق واحد.

تكون احتمالية حدوث فهم خاطئ أكبر عند ضبط عملية تطوير البرمجيات الموزَّعة المراكز. عندما يكون المشروع منتظماً، لا يكون من غير الشائع إجراء مقابلات مباشرة من أجل زيادة المواصفات. ولا تكون هذه المقابلات أكثر فاعلية وحسب، بل إنها تتيح قياس مستوى الاستيعاب لدى الشخص المتلقي أيضاً. وقد ذكرنا سابقاً أن شبكة الحماية لا تكون متوفّرة فقط في مشاريع تطوير البرمجيات الموزَّعة المراكز؛ إذ يعتمد على أساليب ضمان الجودة في مشاريع تطوير البرمجيات الموزَّعة المراكز. نأمل أن يكون قد تكوَّنَ لديك الآن إدراك أفضل عن السبب الذي يجعل الحصول على جودة المنتج في مشاريع تطوير البرمجيات الموزَّعة المراكز أكثر صعوبة.

11-3-3 الاستراتيجيات المتبعة للحصول على الجودة في مجال تطوير البرمجيات الموزَّعة المراكز

في عمليات التطوير الموزَّعة، يُنصح باتباع وسائل الاختبار التلقائي المحوسب للغة البرمجة. سوف تقوم الفرق التي تعمل عن بُعد بعمل الاختبارات مع أعضاء فريقهم المحلي، ويمكن استخدام هذا الأسلوب على شكل فريق ثنائي من المبرمجين. على كل حال، من المفيد في عمليات التطوير الموزَّعة وضع أسس ومعايير للغة البرمجة التي يستخدمها جميع أعضاء فريق البرمجة. إذا تم اختبار البرمجة المستخدمة باستخدام أداة لاختبار وجود اختراقات للأسس والمعايير الموضوعة للغة البرمجة، سيكون من الأسهل على أي عضو من أعضاء الفريق فهم البرمجة الموضوعة وسيتم اكتشاف العيوب قبل تنفيذ عملية الدمج.

نعرّف «أسلوب العمل الممتد» على أنه النموذج الذي تتم فيه برمجة كل المكوّنات عن بُعد ومن ثم يتم جمعها معاً في الموقع المركزي لإجراء عمليتي الدمج والاختبار. لذلك، يتم إجراء عدة مراحل من الاختبارات في الموقع الذي يعمل عن بُعد وفي الموقع المركزي. وتتضمن بعض الممارسات الجيدة للاختبار والتي ينصح بها ما يأتي:

- يقوم الفريق المركزي بعمل مجموعة اختبارات القبول والمستمدة من المتطلبات الوظيفية ذات الأولية القصوى.
- يقوم الفريق الذي يعمل عن بُعد بعمل مستويات متعددة من الاختبار التلقائي كجزء من مهامه في كل مرحلة عمل.
- يقوم الفريق المركزي بتحديد وتطبيق اختبارات الدمج في الموقع المركزي. ويتم تحديد هذه الاختبارات عندما تكون إحدى المراحل على وشك الانتهاء، ويتم تطبيقها في نهاية كل مرحلة.

من الأفكار الجيدة استخدام النماذج الأولية لمختلف الغايات. ويفيد استخدام نموذج أولي لواجهة المستخدم أحياناً، وذلك لتأكيد المتطلبات التي نحتاجها، وللتأكد من أن الفرق التي تعمل عن بُعد قد استوعبت هذه المتطلبات بشكل عام. إضافة إلى ذلك، ننصح بأن يتم برمجة «شريحة عمودية» من هيكلية النظام أولاً، وذلك لتسهيل إجراء اختبارات مبكرة على خصائص المستويات المختلفة للنظام، والتأكد من الجزء الأكثر خطورة من الهيكلية، والتأكد أيضاً من أن الفرق التي تعمل عن بُعد قد استوعبت المتطلبات بشكل كامل وأصبحت قادرة على تطبيق التصميم. قمنا إلى جانب هذه الأمور بإجراء اختبار تحليلي ثابت (من دون تشغيل النظام) على هيكلية النظام وعدد مختلف من الأجزاء البرمجية الأساسية. يمكن أن يتم إجراء ذلك بشكل تلقائي محوسب. سيتم التأكد من أن تفاصيل التصميم قد تم إنجازها بشكل فاعل إذا توفّرت أداة فعَّالة لأداء ذلك.

من المفيد إضافة عملية الاختبارات التي تُجرى لملاءمة ما يتم تسليمه مع المواصفات كجزء من حزمة العمل في نهاية كل مرحلة معيَّنة على الرغم من أنها تشكل عبئاً إضافياً على الفريق المركزي. قد وجدنا أن اتباع أسلوب الاختبارات التي يقوم بها المبرمجون تساعدهم على فهم وتحديد المتطلبات الأكثر أهمية.

نقوم بإجراء سلسلة من عمليات المراجعة للمتطلبات من قبل خبراء في المجال للتأكد من عدم إغفالنا أي متطلبات أساس، وأن المتطلبات كما تم تصورها تعبر عن حاجات ورغبات المستخدم، وذلك كجزء من العمليات التي تجرى على المتطلبات. ويساعد هذا بشكل كبير في سعينا إلى الحصول على النموذج المطلوب. ويؤدي التمثيل المرئي باستخدام لغة النمذجة الموحدة إلى تعبير بسيط بحيث

يتمكن الأشخاص غير التقنيين من فهمه. كما يتم من خلالها إجراء اختبارات تلقائية وتزيد من القدرة على إجراء التعديلات وتغطية مجالات الاختبار.

11-4 صيانة المنتج

ما أن يصبح المنتج جاهزاً حتى يدخل في مرحلة الصيانة كجزء من دورة إنتاج المنتج. في هذه المرحلة، ينتقل التركيز من جودة المنتج إلى إرضاء العميل. ولا يمكن إجراء الكثير في ما يتعلق بجودة المنتج، إذ يتم في هذه المرحلة توزيعه في موقع العميل بوجود الأخطاء التي لم يتم اكتشافها بعد. فكيف يمكننا التأكد من أن العميل لا يشعر بعدم الرضا نتيجة هذه الأخطاء التي سوف يكتشفها ما أن يبدأ باستخدام المنتج؟ ففي حين تعتبر صيانة المنتج من الأمور التي لا بد من تطبيقها لأي منتج برمجي، غير أن إصلاح العيوب الموجودة في منتج تم تطويره عن طريق عدة مزودين في مواقع مختلفة تعمل عن بُعد بفاعلية وكفاءة تمثل تحدياً. في هذا القسم، سوف نلقي نظرة على صيانة المنتج في مشاريع تطوير البرمجيات الموزَّعة المراكز والحاجة إلى العلاقة طويلة الأمد مع المزودين في مواقع البرمجة التي تعمل عن بُعد والاستراتيجيات المتَّبَعَة لتعزيز مثل هذه العلاقات.

11-4-1 صيانة المنتج في إطار شامل

يتم الحكم على جودة صيانة المنتج عن طريق فاعلية وكفاءة عملية الصيانة. من التدابير النموذجية التي وضعت لتقويم ذلك (Kan, 2003) ما يأتي:

- تراكم الإصلاحات: أصبح من الممكن قياس المعدل الذي يتم به إجراء إصلاح العيوب التي يتم الإبلاغ عنها.

- **سرعة الاستجابة لعمليات الإصلاح**: عملية قياس معدل الوقت اللازم لإصلاح جميع الأخطاء من اللحظة التي يتم الإبلاغ بها، إذ يمكن أن يتم تصنيف العيوب ويتم قياس الوقت اللازم للاستجابة اعتماداً على مستويات العيوب.

- **إصلاح التقصير**: عملية قياس الإصلاحات التي تتم ببطء بحيث تأخذ وقتاً أطول من الوقت اللازم للاستجابة.

- **جودة الإصلاحات**: عملية قياس الإصلاحات غير الصحيحة، أي الإصلاحات التي لا تقوم بتعديل العيوب أو تؤدي إلى عيوب جديدة.

بالنظر إلى هذه المعايير، يتضح أن عملية الإصلاحات ستكون فعَّالة وذات كفاءة اعتماداً على فاعلية وكفاءة الأشخاص الذين يقومون بها. وسوف تعتمد فاعلية وكفاءة الأشخاص الذين يقومون بالإصلاحات على مدى معرفتهم الوطيدة بالمنتج الذي يقومون بصيانته؛ ولا يوجد بديل لذلك. ومن الواضح أن الأشخاص ذوي المعرفة الوطيدة بالمنتج هم أنفسهم من قام ببرمجته. فلذلك يجب أن نرغم المزودين في المواقع التي تعمل عن بُعد على أن يقوموا بصيانة المنتج بالإضافة إلى جانب برمجته. ويجب أن نلاحظ أن انتساب المزودين للتنظيم المركزي الذي يمتلك المشروع قد يضيف بعداً آخر لعملية الصيانة. وسوف تختلف الطريقة التي يتم بها التعامل مع أعمال الصيانة عندما يكون كلٌ من التنظيم المركزي والمزودين جزءاً من نفس الكيان قانونيين عن الطريقة التي تستخدم عندما يعملان في كيانين قانونيين مستقلين. وسنلقي مزيداً من الضوء على ذلك في القسم القادم.

11-4-1-1 الحاجة إلى وجود علاقة عمل طويلة الأمد

تمثل عملية بناء فريق عمل تحدياً كبيراً في بيئة العمل الموزَّعة؛ إذ يتطلب ذلك جهداً كبيراً كالسفر لتعريف أعضاء الفريق الداخلي بنظرائهم في الخارج وتكوين موقع وهمي مشترك للمشاركة في التغييرات وبناء الأنظمة الإدارية ومجتمعات التدريبات، حيث يستطيع أعضاء الفريق الحصول على أجوبة عن تساؤلاتهم. لذلك من المفيد المحافظة على علاقة مع الفرق الخارجية لمدة زمنية طويلة، وإن لم تفعل، فيجب أن تبذل الجهد الكبير نفسه في كل مرة تجد فيها شريك عمل جديد.

ومع ذلك، كما ورد في الملاحظات الافتتاحية، يكون بناء تعاون طويل الأمد مع الفريق الخارجي أكثر أهمية للحصول على دعم مستمر وإجراء الإصلاحات على المنتج قيد البرمجة.

يجب أن يتم التعامل مع الفرق الخارجية كامتداد لعمل الوحدة التجارية المركزية التي تمتلك المنتج. لذلك يقوم الفريق المركزي بوضع الخطط لبناء الموقع المختص ليكون مركزاً لأعمالهم بمرور الوقت. وهذا يعني أنه من المفيد تدريب الفريق الذي يعمل عن بُعد على المدى البعيد للعمل التجاري. سوف تتحسن فاعلية التعاون مع مرور الوقت (Lasser and Heiss, 2005). لذلك، سوف يتم لمس العوائد من التكلفة التي تم بذلها على مشروع التطوير الموزع بعد سنوات من العمل التعاوني، أو قد يُلمس ذلك في المشروع القادم، وليس في أول مشروع مع وجود مواقع جديدة.

11-4-1-2 الاستراتيجيات المتَّبعة لتعزيز العلاقات الودية

هناك عدد من الطرق المتاحة لتطوير علاقات عمل جيدة مع المزودين الذين يعملون في المواقع التي تعمل عن بُعد، منها:

- **الثقة**: لن تنجح العلاقات طويلة الأمد من دون وجود ثقة متبادلة. فيجب أن يتم اكتساب الثقة التي تحتاج إلى وقت لبنائها؛ غير أنه يمكن اختصار الفترة الزمنية اللازمة لبناء الثقة عن طريق «الالتزام بعمل ما تقول». من المقترحات لذلك القيام بعمليات التسليم في المواعيد المحددة؛ إذ يجب أن يكون هناك التزام من كلا الطرفين، ويجب الالتزام بالجدول الزمني.
- **احترام الزملاء**: كجزء من استراتيجية بناء مركز منافس يعمل عن بُعد، يجب أن يُعامل أفراد فريق العمل كزملاء ذوي قيمة ومحترمين. وهذا يختلف عن معاملة أفراد فريق العمل في الموقع الذي يعمل عن بُعد كعمال يعملون معك لمدة قصيرة فقط. ويؤدي هذا الاحترام إلى فائدة طويلة الأمد في بناء مهارات تقنية وخبراء في المجال من أفراد الفريق الذي يعمل عن بُعد، هذا على افتراض أن العلاقة سوف تستمر لمدة طويلة.
- **تبادل أفراد فرق العمل**: إن أفضل طريقة لنقل المعرفة وبناء الثقة بين التنظيمات هي جعل تبادل أفراد فرق العمل بين المواقع المختلفة. وللوصول إلى غايتنا في عملية تطوير البرمجيات الموزَّعة المراكز، نشجع انتقال أفراد فريق العمل الذي يعمل عن بُعد إلى الموقع المركزي للمشاركة في أنشطة المراحل الأولية. كما ننصح أن يعمل بعض أعضاء الفريق المركزي كمندوبين لفترات طويلة في موقع الفريق الذي يعمل عن بُعد.
- **تخطيط الكفاءات وتطويرها**: يجب وضع الخطط لأعضاء الفرق التي تعمل عن بُعد حتى ينمو لديهم حس المسؤولية بحيث يصبحون مركزاً تنافسياً. على سبيل المثال، في مشروع نظام إدارة المباني الآلي، تم التخطيط لأن يصبح مركز البرمجة في سلوفاكيا منافساً في مجال المعدات والتطبيقات لمشروع دائرة التلفزة الخاصة (CCTV). كانت البداية بتطوير نماذج العمل

وتطبيقات مشروع دائرة التلفزة الخاصة، غير أنهم أصبحوا خبراء تقنيين في هذا المجال.

- **الحفاظ على تماسك فرق العمل**: بما إن رعاية العلاقات أمر حتمي لتبقى طويلة الأمد، فمن المهم الحفاظ على الأعضاء المميزين لفريق الإنتاج معاً. وحين تتحسن الظروف الاقتصادية في الدول ذات المرتبات المتدنية، قد يتجه مهندسو البرمجيات إلى تغيير وظائفهم باستمرار لتحسين مرتباتهم. ويجب أن يحاول المزودون المحافظة على الفرق التي تعمل لديهم معاً لسنين عديدة لإجراء وتنفيذ مهام التطوير والصيانة. ويمكن أن يتم هذا عن طريق توفير عمل ممتع وطويل الأمد، وأيضاً توفير إغراءات مادية كوجود نظام حوافز لفترات طويلة (LTIP) أو مكافآت دورية.
- **العمل التعاوني**: العمل التعاوني الجيد ليس الأمر الوحيد الذي يجب توفُّره في فريق البرمجة، إذ يجب أن يكون هناك عمل تعاوني جيد بين الفرق التي تعمل عن بُعد والفرق المركزية لعمليات التطوير الموزَّعة. قم باختيار فريق عمل يحقق تواصلاً أفضل بين أعضاء فريقه ويمتلك مهارات اجتماعية. ويمتلك بعض أعضاء فرق العمل موهبة «مهارات الربط»، كالتوفيق بين الخصوم وفهم الثقافات المختلفة والمهارات اللغوية، وغير ذلك.
- **التخطيط لفترة طويلة**: شارك المزودين مخططك طويل الأمد. فعلى الرغم من أن بيئة العمل التجاري قابلة للتغيير، ما يعني ضرورة تغيير الخطط، غير أنه لا بد من وجوب إعلام المزودين بعدد العاملين المطلوب والمهارات الضرورية وأي أمور أخرى تتعلق بالعمل بشكل عام.
- **التفاوض حول السعر**: لا تقم بالضغط على المزودين مادياً حول سعر ساعة العمل، أو أي أسعار أخرى لتوفير مبلغ صغير من

المال. أوصينا في الفصل التاسع بأن يكون تقدير أجرة برمجة أي من أجزاء النظام بمعدل أكبر قيمة وأقل قيمة تقديرية يتم قبولها. وتكون أجور العمالة لدى المزود بحدود 20 بالمئة إلى 30 بالمئة من أجرك. وقد يؤدي الضغط للحصول على خصم مقداره من خمسة بالمئة إلى عشرة بالمئة إضرار العلاقة مع المزود أكثر من الفائدة المحرزة من التوفير في النقود.

فكرة مفيدة:

قم بمراقبة الأشخاص المسؤولين عن المشتريات خلال فترة التفاوض حول العقد مع المزودين.

يوجد عدد كبير من أعضاء فريق العمل ممن يمتلكون الحافز لتقليل التكاليف بنسب مئوية محددة من قيمة العقد. في الكثير من الحالات، قد لا يدركون (أو لا يهتمون) الأهداف المتعلقة بالعقد الخاص بالمشروع أو العمل التجاري. قم بإقناع أعضاء فريق المشتريات أن التوفير في التكاليف لم يجدِ شيئاً إذا لم ينجح البرنامج عند تسليمه.

11-4-1-3 انتماء المزودين إلى المؤسسة المركزية

عندما لا يكون المزودون جزءاً من نفس الكيان القانوني الذي ينتمي إليه التنظيم المركزي، من المرجح أن يكونوا جزءاً من وحدات تنظيمية أخرى ويمتلكوا أساساً مختلفاً أو شركات مستقلة. ما أن يتم إنهاء المشروع البرمجي، ينتقلون إلى مشروع آخر. في ظل هذه الظروف، من المهم أن يُوجد الفريق المركزي استراتيجية لتحديد الأولويات وتحديد التغييرات التي يطلبها العملاء عند تحديد الأعضاء المناسبين في التنظيم الخاص بالمزود. قد يؤدي هذا إلى إشكالية، حيث إنه قد يكون لكل وحدة تنظيمية أهداف

وأهداف عمل قد تتعارض مع الوحدة الأخرى. إن الاستراتيجية الناجحة هنا هي وجود عقد صيانة مع الفرق التي تعمل مع المزود وعرض حوافز إضافية إذا عملوا معك في مشاريع مستقبلية. كما تم مناقشته في القسم السابق، الهدف هو أن تتم معاملة المزود على أنه سيقدِّم عملاً مستمراً وتقوم بتطويره بحيث يصبح مركزاً منافساً. لذلك، يجب على التنظيم المركزي أن يجعل المزودين جزءاً من استراتيجية العمل المستقبلي لديهم.

إذا كان المزودون عبارة عن كيانات قانونية مستقلة، يجب أن يحذر التنظيم المركزي من تغيّر الموظفين إذا كان المزود موجوداً في منطقة مضطربة. لاحظنا وجود معدل تغيّر عالٍ في موظفي الشركات التي توجد في مدينة بانغالور في الهند. قد تساعد الاستراتيجيات التي تم التطرق لها في القسم السابق في تقديم حوافز طويلة الأمد في مثل هذه الظروف.

11-5 الملخص والاستنتاجات

من المهم استخدام الممارسات المتعلقة بضمان الجودة للحصول على عوائد ناجحة في مشروع التطوير الموزع. يجب أن يشارك العاملون في مجال ضمان الجودة في الموقع المركزي زملاءهم في المواقع التي تعمل عن بُعد للمساعدة في التحكم وتحسين عمليات البرمجة. يتم منع ظهور مشاكل متعلقة بالجودة بشكل أفضل، ويجب أن يتم اكتشافها قبل تسليم البرامج للعميل.

11-6 أسئلة للمناقشة

1. ناقش بعض المعايير والاستراتيجيات المتبعة للحفاظ على العمليات وجودة المنتج في مجال تطوير البرمجيات الموزَّعة المراكز.

2. ما هو هدف المحافظة على علاقات ودية طويلة الأمد مع المزودين في المواقع التي تعمل عن بُعد؟
3. ما هو طول الفترة الزمنية المتوقعة واللازمة للفريق الذي يعمل عن بُعد للوصول إلى أفضل أداء؟
4. كيف يعمل خبير الجودة في الموقع المركزي مع الخبراء في المواقع التي تعمل عن بُعد؟

المراجع

Books

Hofmeister, Christine, Robert Nord and Dilip Soni. *Applied Software Architecture*. Boston; MA: Addison-Wesley, 2000.

Kan, Stephen H. *Metrics and Models in Software Quality Engineering*. Boston; MA: Addison-Wesley, 2003, pp. 105-110.

Moeller, Karl-Heinrich and Daniel J. Paulish. *Software Metrics: A Practitioner's Guide to Improved Product Development*. Los Alamitos, CA: IEEE Computer Society Press, 1993.

Paulish, Daniel J. *Architecture-Centric Software Project Management*. Boston; MA: Addison-Wesley, 2002.

Conferences

Lasser, S. and M. Heiss. «Collaboration Maturity and the Offshoring Cost Barrier: The Trade-Off between Flexibility in Team Composition and Cross-Site Communication Effort in Geographically Distributed Development Projects.» *Proceedings of the IEEE International Professional Communication Conference (IPCC 2005)*, Limerick, Ireland, 10-13 July 2005, pp. 1-11.



الفصـــل الثاني عشر

دعم البنية التحتية في تطوير البرمجيات الموزَّعة المراكز

لقد حضرت للتو أولى الفرق الموزَّعة إلى مشروع الأستديو العالمي من أصل ست فرق، وكان زاك يحاول وضع أسس لدعم البنية التحتية، لا تحقق المطالب التي تحتاجها التقنية والبنية الهندسية للمشروع وحسب، بل وأيضاً الطبيعة التي تنتشر بها جميع الفرق. وكان الأمر الأساس موضع الخلاف في الفريق المركزي الطريقة التي تمكنّا من تسهيل التواصل بين الفرق وكيفية التأكد من توفُّر مهارات حديثة للمشروع في التنظيم. بعد الانتهاء من وضع إدارة لعملية إنشاء البرمجيات، يحتاج مسؤول عمليات البناء البرمجي إلى تحديد طريقة للتعامل مع أجزاء العمل البرمجي ودمجها على أساس المواقع المتعددة التي تقوم بالبرمجة.

نلقي في هذا الفصل الضوء على الاستراتيجيات والأدوات المستخدمة في إدارة البنية التحتية في بيئة تطوير البرمجيات الموزَّعة

المراكز، ويوفِّر دليل استخدام عملي للطريقة الأفضل التي يمكن بها إدارة ذلك. ونغطي في هذا الفصل تحديداً النواحي الآتية:

1. المعايير التي يتم من خلالها اختيار البنية التحتية في مجال عمل تطوير البرمجيات الموزَّعة المراكز.
2. البنية التحتية المطلوبة للاتصالات والتنسيق، وإدارة المعرفة، وإدارة عملية تكوين البرمجيات.
3. العمليات المستخدمة لتسهيل استخدام البنية التحتية بشكل مناسب وفعّال.

12-1 معايير اختيار البنية التحتية

يعتبر التحضير لبنية تحتية تُلبِّي حاجات المشروع لعمليات التطوير الموزَّعة من العوامل المهمة لنجاح المشروع جنباً إلى جنب مع الأنشطة الهندسية والتخطيطية. ويجب أن تدعم أدوات البنية التحتية سهولة الوصول والتعاون والتوافق وسير العمليات والمعرفة والتكامل. وقد لا تحقق الأدوات التقليدية هذه المتطلبات بشكل جيد؛ وإذا تم التنبؤ بذلك في وقت متأخر، يؤدي انتشار هذه الأدوات التقليدية وهجرة الكفاءات إلى هدر المال والوقت، وفي نهاية الأمر إلى عدم سير المشروع كما يجب (Gersmann, 2005).

12-1-1 القدرة على الوصول

يجب أن تكون الأدوات متوفّرة في جميع مواقع البرمجة. وذلك يعني أنه يجب أن تتوفّر لدى أعضاء فريق المشروع في المواقع المختلفة الصلاحيات الملائمة. إضافة إلى ذلك، يجب معالجة أي مشاكل تتعلق بشبكة الربط. وقد يكون هناك تعارض بين المواقع اعتماداً على شبكة الربط، وقد يسبب ذلك صعوبة في تبادل المعلومات والتوصيل بين الشبكات. إذا كانت هذه الأدوات تعتمد

على شبكة الربط، يجب أن تدعم هذه الأدوات عملية التوصيل في بيئة شبكات مقيدة الصلاحيات. إذا كان النظام الذي يتم تطويره نظاماً موزعاً، فقد تتأثر الشبكات التي يحتاجها النظام. فيجب أن يقوم مسؤول تنسيق البنية التحتية بتوثيق ذلك ويرسل تقريراً إلى المعنيين حول هذه الأمور. وقد يعني ذلك أن بعض الأدوات قد لا تكون مؤَهَّلة للانتشار في المشاريع الموزَّعة، ولذلك لا يمكن استخدامها. وقد تحتاج الأدوات التي تدعم عملية الانتشار وشبكات الربط بشكل محدود إلى دراسة إمكانيتها (على سبيل المثال، الحلول المبنية على أساس شبكة الإنترنت، أو الأدوات التي تستخدم بروتوكول نقل النص التشعبي HTTP يجب أن يقوم مسؤول تنسيق البنية التحتية بإجراء التعديلات المطلوبة على إعدادات الشبكة أو على تنفيذ البنية التحتية).

12-1-2 التعاون والتوافق

يجب أن تدعم الأدوات التعاون والتوافق وفقاً لطبيعة مجال التطبيق. ولا يشير التعاون بالضرورة إلى تعاون متزامن، ولكن يشير إلى تعاون غير متزامن أيضاً، وهو التعاون المطلوب بين الفرق الموزَّعة جغرافياً في عدة دول يكون التوقيت فيها مختلفاً. ويشير مصطلح التوافق إلى المدى الذي يمكن به أن تتعاون الفرق المختلفة فعلياً أو ضمنياً في إحدى المهام المطلوب إنجازها. ويمكن أن يكون دعم التوافق الضمني على هيئة تفعيل عمليات الدخول والدمج.

يمكن أن يتم توفير هذه الوظائف عن طريق تعديل نظام التحكم ليصبح على شكل أدوات تعمل على أساس دمج النصوص أو بواسطة التطبيقات نفسها لتصبح على شكل أدوات دمج محددة. يتطلب التحكم بعملية التزامن البحث في المهام أو الأجزاء الصغرى

منها بهدف إجراء التعديلات. تحتاج هذه الأنظمة عادة إلى أسلوب عمل باستخدام شبكة الإنترنت. يجب أن تكون أجزاء المهام التي يتم التعامل معها أصغر ما يمكن، ومن ثم يتم تجميعها إلى أجزاء أكبر، وذلك لتقليل احتمال وجود أي تعارض. ومن المهم أن تكون عملية دعم التعاون والتوافق ذات مستوى متقدم، نظراً إلى أنها تتحكم بسهولة الاستخدام، وبالتالي احتمالية وجود اعتماد كافٍ من قبل أعضاء الفريق.

12-1-3 العمليات

هناك أمران مهمان مرتبطان باختيار البنية التحتية، وهما المدى الذي تتحكم فيه الأدوات بتنفيذ عملية معينة أو أن يكون هناك تعريف مرن للعملية. لسوء الحظ، من الصعب إيجاد أداة تدعم كلا الأمرين، إذ كلما وفَّرت الأداة مرونة أكبر، كان من الصعب فرض تنفيذ عملية بواسطة وسائل تقنية. فعادة ما تكون الأدوات التي توفّر بنية قوية وسلطة على تنفيذ العمليات مصممة لتخدم عمليات ومخططات سير عمل محددة، لذلك توفّر القليل من المرونة. يجب على مدير المشروع ومنسق البنية التحتية أن يتخذا قراراً بخصوص الأدوات التي يجب استخدامها وفقاً لطبيعة المشروع. في مجال مشاريع التطوير الموزَّعة، تفضّل الأدوات الأكثر مرونة أكثر من الأدوات المحددة، لأنها نادراً ما تكون مصممة من أجل مشاريع التطوير الموزَّعة. على كل حال، يؤدي اختيار أدوات مرنة إلى حاجة أكبر إلى أداء يدوي لفرض تنفيذ العمليات.

12-1-4 المعرفة والتكامل

يجب أن تدعم البنية التحتية المعرفة بالتواصل والمهام التقنية. على سبيل المثال، يمكن تدعيم المعرفة بتوفير أكبر قدر منها في

موقع واحد وربط المهام المختلفة بالموقع ليتمكن المستخدمون من استعراضها. عند استخدام الوثائق التقليدية، يصعب الرجوع إلى أي معلومات تتضمنها هذه الوثائق. ومن المفضل امتلاك القدرة على الوصول بشكل مباشر إلى أي جزء من أجزاء المعلومات، وذلك لأسباب تتعلق بالقدرة على التتبع. ويتوفّر لدى الكثير من الباعة مجموعة من الأدوات المدمجة التي تتيح القدرة على التتبع وتصفُّح المعلومات. وتكمن المشكلة في أن التكامل يكون محدوداً بتوقعات مزودي هذه الأدوات. لذلك، يجب أن يتم إجراء عمليات شراء مناسبة، قد تكون بعض الأدوات مصنوعة بشكل أفضل وتدعم إجبار إجراء العمليات، وأخرى قد تكون أكثر مرونة وأكثر سهولة في دمجها مع النظام. ويجب أن يتم تنفيذ الأدوات بحذر ووفقاً لاعتبارات صارمة تحددها متطلبات المشروع (Gersmann, 2005) بحيث تصبح استثماراً مستقبلياً طويل الأمد في البنية التحتية والأدوات.

12-2 التواصل والتنسيق

يسبب نقص البنية التحتية التي تدعم الحاجة إلى طرفي التواصل المتكاملين في مشاريع تطوير البرمجيات ضعفاً في قدرة المبرمجين والإداريين على العمل كفريق متماسك (Herbsleb and Moitra, 2001) (التواصل الرسمي كتحديث حالة المشروع الحالية والمحادثات العابرة غير الرسمية بين أعضاء الفريق). وهناك بُعدان لهذه المشكلة على الأقل (Gersmann, 2005). **أولاً**، هناك مشاكل تتعلق بسياسات البنية التحتية للاتصالات واستخداماتها. **ثانياً**، أثبتت عمليات التواصل أنه من الصعب إعداد وتنفيذ البنية التحتية، حيث تكون عرضة للخطأ ويكتنفها الغموض وتؤدي إلى نقص في المعرفة.

تمثل كلتا المشكلتين تحدياً لمشاريع تطوير البرمجيات الموزَّعة المراكز. في المراحل المبكرة لتطوير البرمجيات، نحتاج إلى عمليات اتصال مكثَّفة (Perry [et al.], 1994). لكن من الصعب المحافظة على محادثات غير رسمية مستمرة بين أعضاء الفرق الموزَّعة بسبب بُعد المسافة وأمور تتعلق بفارق التوقيت، ما يؤدي عادة إلى اختلال في الأنشطة وقد يتطلب الأمر إعادة العمل. قد يكون هناك حاجة إلى إعادة التواصل، هذا سيؤدي إلى وجود ضغط إضافي عندما يضطر الفريق المركزي إلى أن يجيب عن نفس الاستفسارات. وتُستخدم أساليب وأجهزة مرئية مختلفة للاتصال بين أعضاء الفريق الواحد في أوقات مختلفة، وبالتالي يُفقد التناسق في التواصل.

12-2-1 استراتيجية التواصل والتنسيق

تتمثل الدوافع الرئيسة وراء إيجاد استراتيجية للتواصل والتنسيق في الرغبة في زيادة القدرة على الوصول إلى المعلومات من قبل أصحاب العمل بطريقة منضبطة وإنشاء بيئة للمشاركين بحيث يشارك جميع أصحاب العمل في عملية تبادل المعلومات.

يجب أن يكون هناك آلية فعالة للتعامل مع نوعين من التواصل، معلومات عابرة (معلومات ذات تأثير محلي ومعلومات ذات تأثير قصير الأمد) ومعلومات مستمرة (المعلومات التي لها تأثير مهم في الفرق المختلفة أو التي يجب أن يتم توثيقها ليكون بالإمكان الرجوع إليها في ما بعد). وقد يكون البريد الإلكتروني طريقة مفيدة للحصول على المعلومات العابرة، لأنه طريقة سريعة ومن السهل استخدامها، لكنه لا يناسب المعلومات المستمرة بسبب عدة عوامل، إذ لا يمكن استخدامه في المعلومات الأساس، أو لضمان مستوى ثابت للحصول على المعلومات من قبل أصحاب العمل، أو لضمان توفُّر المعلومات حين احتياجها. ولن يكون البريد الإلكتروني مصدر معلومات مثالياً

لاستعادة المعلومات بعد استقبالها بأيام أو أسابيع بسبب الضعف في البنية والتنظيم. ولتحقيق انخفاض ملحوظ في نفقات التواصل على المدى البعيد، يجب أن يكون هناك قدرة على الوصول للمعلومات دون الحاجة إلى اتصالات أو استعلامات معقدة. ومن المهم أن يكون التواصل والوثائق واضحة وضوحاً تاماً. وعلى أعضاء الفريق المركزي أن يلعبوا دور الميسِّر لهذا بدلاً من دور أصحاب الصلاحيات المطلقة (Gersmann, 2005).

12-2-2 البنية التحتية الخاصة بالتواصل والتنسيق

يجب أن يتم تصميم البنية التحتية الخاصة بالتواصل والتنسيق بعناية بحيث تدعم الأهداف الأساس الموضوعة لها، وهي سهولة الوصول والتعاون والتوافق ودعم إجراء العمليات والمعرفة والتكامل. سوف نقوم في الأقسام الآتية بوصف البنية التحتية التي أثبتت جدارتها من خلال خبرتنا.

12-2-2-1 قوائم عناوين البريد الإلكتروني

يمكن أن يتم العمل على التواصل عن طريق البريد الإلكتروني باستخدام قوائم لعناوين البريد الإلكتروني، وهو ببساطة إرسال الرسائل الإلكترونية لجميع الفرق والتأكد من وجود عناوين لجميع أعضاء الفرق. ويتيح استخدام قوائم عناوين البريد الإلكتروني أيضاً الفهرسة الآلية لهذه العناوين في فهرس خاص، وبالتالي حفظ هذه العناوين. ويجب أن يتم نسخ جميع عناوين أعضاء الفريق المركزي في موقع أو لدى كل فريق (Gersmann, 2005). يجب أن تتوفّر أيضاً قوائم عناوين لدى الفريق المركزي، لأن ذلك يسهِّل تفاعل فريق المصممين والرؤساء ذوي العلاقة بكل فريق. وبافتراض أن استخدام الرسائل الإلكترونية هو الأكثر شيوعاً للتواصل، فإن قوائم العناوين تساعد في تبسيط التعامل مع الرسائل الإلكترونية.

12-2-2-2 البنية التحتية المتعلقة بالاجتماعات الأسبوعية

يجب أن يعقد الفريق المركزي اجتماعات إدارية أسبوعية مع كل فريق من الفرق التي تعمل عن بُعد. ولتسهيل ذلك يمكن استخدام مقاسم خاصة بالاجتماعات أو أجهزة محادثة متلفزة أو المحادثة الصوتية أو أجهزة ربط الحواسيب الخاصة بالأعضاء. تتيح مقاسم الاتصال التواصل مع عدة أعضاء موزعين، وفي نفس الوقت تجعل هذه الخدمة الفرق قادرة على استخدام رقم هاتف محدد ورمز للقاعة التي يجري فيها الاجتماع للمشاركة في المحادثة. كما إنها تمتلك خصائص أخرى مثل الحصول على محادثة واضحة من دون تشويش. ويمكن استخدام المحادثة المتلفزة في بداية الاجتماع مع المواقع التي تعمل عن بُعد وفي حالة الاجتماعات غير العادية التي تتطلب ذلك. ويمكن استخدام المحادثة الصوتية في حال عدم توفّر هواتف داخل القاعات لدى الفريق الذي يعمل عن بُعد. ويجب ربط أجهزة الحاسوب للتأكد من أن كلا الطرفين (الفريق المركزي والفريق الذي يعمل عن بُعد) لديهما نفس التصور حول الأمور (المهام التي تم الإشارة لها خلال عملية التحضير لهذه الاجتماعات).

فكرة مفيدة:

أطلب من الفريق الذي يعمل عن بُعد تجهيز مجموعة من الأسئلة كتابياً قبل الاجتماع وإرسالها.

عن طريق الطلب من الفريق الذي يعمل عن بُعد كتابة استفساراتهم وأسئلتهم وإرسالها، يحرص الفريق المركزي على توفير أفكار وأجوبة منسقة للفريق الذي يعمل عن بُعد. ويتيح ذلك الفرصة للفريق المركزي للتفاعل الداخلي مع أعضاء الفريق الذي يعمل عن بُعد وإزالة أي غموض أو اختلافات مستقبلية. تساعد الإجابة عن الأسئلة

والاستفسارات في إزالة أي غموض ينتج عن الجدال في الاجتماع مع الفريق الذي يعمل عن بُعد والغموض الذي ينتج من الحاجة إلى تغيير أو تعديل. كما ستؤكد أن يكون أعضاء الفريق المركزي الذين لم يشاركوا في الاجتماع على معرفة بالأسئلة الموجهة ومستوعبين لإجاباتها. كما نوصي أن يتم إرسال جميع الأسئلة الجيدة وإجاباتها إلى مكان ما بحيث تستطيع جميع الفرق الوصول إليها.

12-2-2-3 استخدام منتديات المناقشة للمباحثات التفاعلية والاستفسارات

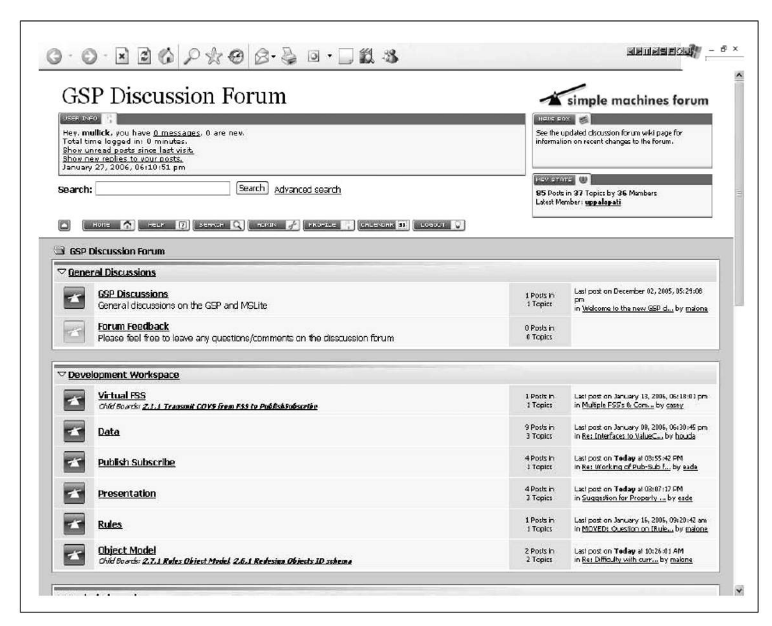
يستطيع أعضاء الفريق الذي يعمل عن بُعد التسجيل في منتديات المناقشة الخاصة بمواضيع مختلفة مترابطة كإحدى الطرق التي يمكن من خلالها الحصول على المعلومات من الفريق المركزي. وتضمن هذه المنتديات فهرسة الاستفسارات، واستخدامها متاح دائماً غير محدود بإطار زمني ويسمح لكل الفرق التي تعمل عن بُعد الوصول إلى استفسارات بعضهم بعضاً. وينتج من ذلك تخفيف ضغط العمل عن الفريق المركزي. يُظهر الشكل 12-1 لقطة لأحد منتديات المناقشة الخاصة بمشروع الأستديو العالمي؛ يمكنك الإطلاع على مزيد من التفاصيل حول هذا الموضوع في الفصل الرابع عشر من هذا الكتاب.

فكرة مفيدة:

تأكد من استيعاب الفريق الذي يعمل عن بُعد للتعليمات.

إن إحدى الطرق التي تساعد على ضمان أن الفريق الذي يعمل عن بُعد قد استوعب بشكل كامل التعليمات أو

إجابات الأسئلة هي عن طريق جعلهم يعيدون بكلماتهم الخاصة الإجابات التي فهموها. ويمكن أن يتم هذا شفوياً، أو كتابياً عن طريق جعلهم يكتبون السؤال والإجابة خلال الاجتماع، أو إرسال السؤال والإجابة إلى منتديات المناقشة. إن الخيارين الأخيرين لهما ميزة أن التساؤلات وإجاباتها يمكن أن تصل إليها الفرق الأخرى التي تعمل عن بُعد.

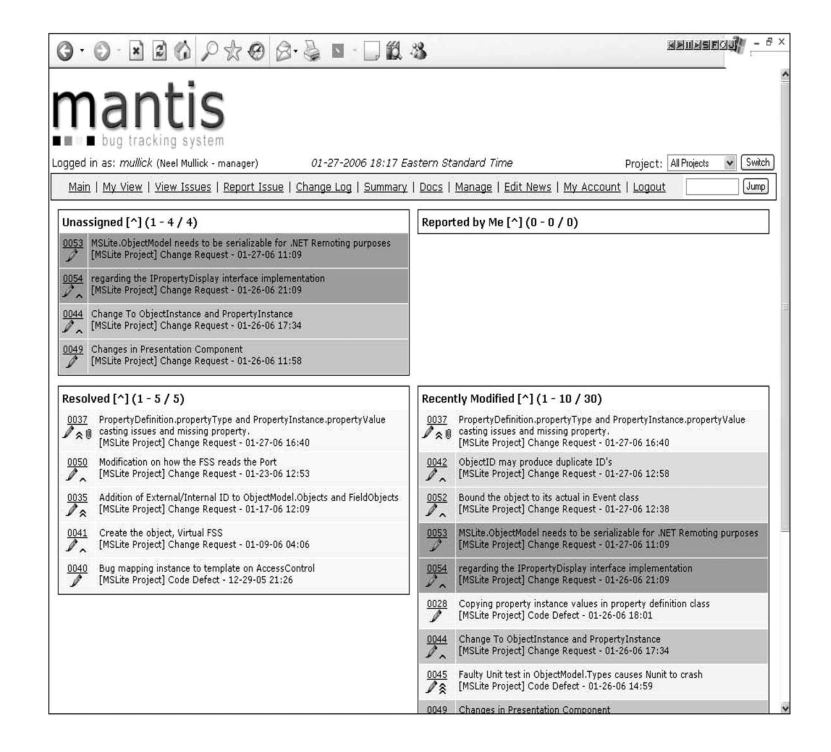


الشكل 12-1: منتدى مناقشة لمشروع الأستديو العالمي

12-2-2-4 تتبع عيوب البرنامج وإدارة التغيير

يجب توفير أدوات مناسبة لتتبع العيوب وإدارة التغيير كما هو الحال في منتديات المناقشة، إضافة إلى ضرورة وجود عمليات تدعم وتقوم بتتبع العيوب وإدارة التغيير (انظر الشكل 12-2).

من المهم ملاحظة أن جميع الإعدادات اللازمة للبنية التحتية التي نوقشت في هذا الفصل تقوم على أساس استخدام الأدوات ذات المصادر المفتوحة(*) (open source) (بعد إجراء بحث وتحليل حول المعايير التي تم وصفها سابقاً) لمشروع الأستديو العالمي، وذلك لتجنب أو التحايل على تحدي توفُّر المعلومات.



الشكل 12–2: برنامج لكشف أخطاء البرمجة (Mantis)

(*) ممارسات متبعة في عالم البرمجة وتطوير البرمجيات وهي تتيح الوصول إلى الأساسيات الأصلية لمنتج برمجي - مثل الشيفرة البرمجية. ومن الأمثلة عليها برنامج (Asterisk) المخصص للاتصالات الهاتفية من خلال الإنترنت و(Sun Java System Portal Server).

12-3 إدارة المعرفة: تصميم البرمجيات والنماذج والتوثيق

إن عدم وجود آلية فعَّالة لمشاركة المعلومات وضعف صيانة تصميم البرمجيات وعملية التوثيق المرتبطة بها والتعاون في إنجاز المهام يؤدي إلى تفاقم الأمور المتعلقة بإدارة المعرفة بطريقة تسخّر الإمكانيات الفعلية لمشاريع تطوير البرمجيات الموزَّعة المراكز. وقد تؤدي بعض الأمور التقنية كذلك إلى تفاقم مشكلة إدارة المعلومات مثل التداخل في الشبكات وصيغ البيانات غير المتوافقة ووجود إصدارات مختلفة للأدوات في المواقع (Herbsleb and Moitra, 2001). وثمة تحدٍ جديد يضاف إلى عملية إدارة المعرفة الفعالة ناتج من كون طبيعة معظم آليات التواصل غير متزامنة لأنها تسُبِّبُ تأخيراً في الأداء وغموضاً في الرسائل ووجود تكرارات وتقادم في عملية التواصل. ويؤدي هذا أيضاً إلى تكوّن نموذج ذهني لدى أعضاء المواقع التي تعمل عن بُعد مختلف عنه لدى الفريق المركزي، ويختلف أيضاً بين الأعضاء أنفسهم. وبالتالي تكون عملية تطبيق النظام مختلفة عمَّا تم تصميمه.

هناك ثمة ملاحظة مهمة جداً من واقع خبرتنا العملية تفيد بأن عدم وجود بنية تحتية في الموقع المركزي يقود المواقع الأخرى إلى تأسيس بنية تحتية خاصة بها. ولا يخفف إنشاء بنية تحتية معزولة من مشكلة إدارة المعرفة. على الفريق المركزي أن يقوم أيضاً بإعداد دليل عن أساسيات استخدام البنية التحتية بنجاح وإلزام العمل بهذا الدليل.

12-3-1 اختيار البنية التحتية لإدارة المعرفة

يجب أن تركِّز العمليات والسياسات المتَّبَعَة في إدارة المعرفة على بناء قاعدة معلومات يتشارك ويتعاون فيها الجميع. ويجب أن

تتكون قاعدة المعرفة من جميع المعلومات المتعلقة بالمشروع، مثل وثائق المهام التقنية والمهام الإدارية والعمليات والممارسات والبنية التحتية، وغير ذلك. يجب أن يدرك أصحاب المشروع حقيقة أن الإنشاء المجرد للوثائق ليس ذا قيمة بحد ذاته، لكنه وسيلة لتزويد جميع الأعضاء بالمعلومات التي يحتاجونها.

لذلك يجب أن تعمل السياسات على تحفيز التواصل غير المباشر بحيث يصبح استخدام قاعدة المعرفة كوسيلة اتصال مباشرة تركز على الأهداف ذات المدى القصير والمفيدة في المجال الحالي أمراً مفضلاً. ولرفع مستوى المعرفة، يجب أن توفّر قاعدة المعرفة آليَّةً ذكيةً لوضع الملاحظات بحيث يطّلع المهتمون على آخر التعديلات أولاً بأول. كما يجب أن يتم إنشاؤها بشكل منطقي بحيث توفُّر سهولة في التنقل وتجميع المعلومات المتعلقة بعضها ببعض في حقول معينة.

إن وجود قاعدة المعرفة لا يلغي الحاجة إلى اتصالات مباشرة بالطبع. ومع ذلك، يجب أن يكون الفريق المركزي حريصاً على مراقبة عدم توظيف التواصل المباشر كبديل لأنشطة التوثيق. وسوف يعمل التواصل المباشر كآلية للأنشطة المتعلقة بالتنسيق في عملية إنشاء قاعدة المعرفة الخاصة بالمشروع. على سبيل المثال، إذا ظهرت أي أمور أو أسئلة خلال الاجتماعات الأسبوعية أو الاجتماعات غير المخطط لها، يجب أن يكون ذلك حول البند قيد البحث من قبل الفريق المركزي أو الفريق الذي يعمل عن بُعد، وبالتالي يجب أن يظهر في قاعدة المعرفة. ويجب على الفريق المركزي أن يراقب المعلومات ويلتزم بصيانتها الفريق الذي يعمل عن بُعد.

يجب أن تُجزأ قاعدة المعرفة إلى أقسام عامة أو شاملة، وأقسام

شبه عامة أو أقسام لفريق محدد. وبذلك يمكن المحافظة على الاستمرارية والتناسق والوضوح على الرغم من السماح للفرق بالمشاركات الداخلية باستخدام نفس الآلية التي تُستخدم في الشركات العامة. ويجب عدم التهاون مع الأقسام المنفصلة لأنها لا تؤدِّي إلى الشفافية في المعلومات بل تحرم الفرق الأخرى من الوصول إلى هذه المعلومات، وبالتالي الحد من المعرفة. وبما إنه قد يلزم المشروع استخدام وثائق مهيكلة وأخرى شبه مهيكلة، لذا يجب استخدام تقنيات وأدوات مختلفة لأداء ذلك. بشكل عام، يجب أن يؤخذ بعين الاعتبار أنه من المفيد الاحتفاظ بأكبر قدر من المعلومات مركزياً باستخدام أداة واحدة لتحسين القدرة على الوصول إلى المعلومات. إن توفير معلومات عامة عن مهام المشروع الرئيسة للمهتمين بالمشروع من شأنه دعم التنسيق ويوفر التوجيه والإرشاد للفريق، وبالتالي توفير تعريف شبه رسمي لطبيعة المنتج المتوقع إنتاجه. يجب أن ينصبّ التركيز الأكبر للفريق المركزي على الالتزام بسياسات وعمليات التواصل خصوصاً في مرحلة التأسيس. يجب أن توكل مهمة إدارة التوثيق إلى عضو واحد في كل فريق يعمل عن بُعد ويعيّن مسؤولاً عن التطبيق المقبول للسياسات والعمليات المتعلقة. وبسبب التأخير في التواصل غير المباشر وغير المتزامن، فإن حل هذه المشاكل لا يكون فورياً. وفي مثل هذه الحالات، يجب استخدام البريد الإلكتروني أو المحادثات الهاتفية (Gersmann, 2005).

نلخصّ في ما يأتي النواحي المهمة في إدارة المعرفة التي يجب أخذها بعين الاعتبار عند إنشاء البنية التحتية في مشاريع تطوير البرمجيات الموزَّعة المراكز:

1. إنشاء قاعدة معرفة مركزية ومتماسكة وتوفير أكبر قدر ممكن من المعلومات فيها.

2. إنشاء عمليات اتصال غير مباشرة وإلزام الأعضاء باستخدامها.
3. توفير أسس ووسائل لعمليات التوجيه.
4. استحداث وظيفة مدير توثيق في كل فريق يكون مسؤولاً عن الحفاظ على بنية قاعدة المعرفة.

12-3-2 البنية التحتية لإدارة المعرفة

إن المحور الأساسي للبنية التحتية لعملية إدارة المعرفة هو مفهوم قاعدة المعلومات، كما ذكرنا سابقاً. تحتاج متطلبات هذه القاعدة للمعلومات تطبيقاً تم إنشاؤه اعتماداً على شبكة الإنترنت، خصوصاً أنها يجب أن تدعم التوفُّر المستمر في بيئة شبكات مقيدة وتدعم أيضاً التعاون والتزامن. الفائدة من هذا التطبيق أنه يُسهِّل عملية نشر المعلومات، والقدرة على الوصول إلى المعلومات باستخدام المتصفحات الشائعة، ويتيح إمكانية الاستخدام في أي مكان عن طريق بروتوكول نقل النص التشعبي (HTTP) المتوفّر في جميع أنظمة الشبكات. تظهر التعديلات التي تُجرى على مصادر المعلومات الموجودة في الإنترنت فوراً لجميع المهتمين بالمشروع؛ ويمكن الوصول إلى إحدى المهام عن طريق اختصار شائع واحد (في حال وجود 'WWW'، فإن هذا الاختصار هو عنوان الموقع الإلكتروني على الإنترنت 'URL'). إن ما يقيد ذلك هو أن المعلومات تكون متوفّرة إذا كان المستخدم متصلاً بشبكة الإنترنت فقط. أحياناً، يكون سير العمل غير مرضٍ لدى تطبيقه باستخدام أدوات تعمل من خلال شبكة الإنترنت، ولا تتوفّر أدوات مناسبة لجميع المهام التي تتطلب تنفيذ من قبل المهتمين بالمشروع. لذلك قد يكون من الضروري دمج الأدوات التي تعمل من خلال شبكة الإنترنت في أدوات تقليدية خاصة بالأجهزة التابعة فقط أو خاصة بالأجهزة التابعة والخادمة.

ومن أدوات إدارة المحتويات البسيطة التي اكتسبت شعبية مؤخراً ما يعرف بالـ Wiki(*). وتوفُّر هذه الأدوات أنظمة تتيح معالجة صفحات الإنترنت بشكل فوري في نافذة المتصفح عن طريق توظيف جمل برمجية خاصة يمكن استخدامها وتعلُّمها بسهولة. هذه الأدوات مناسبة للوثائق شبه المنتظمة. إذ يمكن إنشاء وثائق جديدة بسهولة؛ كما يمكن الإشارة إلى الوثائق المتوفّرة في أدوات الـ Wiki باستخدام جمل برمجية خاصة، أما الوثائق الموجودة في صفحات أخرى خارج نطاق الـ Wiki فيمكن الإشارة إليها عن طريق عنوان الموقع الإلكتروني على الإنترنت بـ (URL). يمكن تضمين الصور وأي أمور أخرى داخل هذه الصفحات.

على الرغم من أن خيارات التصميم والإعدادات ليست غنية كما هي في الأدوات التقليدية، تكون إمكانيات إنشاء وثائق تقنية فعالة جداً في الـ Wiki. وليس بالضرورة أن تكون الوثائق متسلسلةً بدقة، ولكن يمكن تجزئتها إلى أجزاء منطقية بحيث يتم حفظها وتعديلها في صفحات منفصلة. يكون هذا ضرورياً على وجه الخصوص للمتطلبات والهيكلية وخطط المشاريع لأن المهام تقسم إلى أجزاء أصغر ويمكن تتبعها والوصول إليها عن طريق إعادة جمع هذه الأجزاء.

يجب أن يوظف المشروع أدوات أكثر تعقيداً؛ فقد يلزم توفّر خاصية التصدير في هذه الأدوات واستخلاص عروض لصفحة الإنترنت من الوثائق. عند توظيف نظام تحكم عمليات المراجعة،

(*) الـ Wiki هو موقع إلكتروني يسمح بإنشاء وتعديل أي عدد من الصفحات الإلكترونية المترابطة من خلال متصفح إنترنت، وذلك باستخدام لغة ترميز أو أداة تحرير نصوص من نوع «ما تراه هو ما تحصل عليه «WYSIWYG»». ومن الأمثلة على المواقع التي تبنى باستخدام الـ Wiki الصفحات الشخصية المخصصة للتدوين ونظم إدارة المعرفة.

يُسمح بالدخول عن طريق الإنترنت إلى الأجزاء التي يتم التحكم بها، ويمكن إضافة هذه الصفحات إلى نظام المراجعة ويتم الإشارة إليها عن طريق الوثائق الأساس التي تم الاحتفاظ بها في أداة الـ Wiki. كما قد يكون من الممكن الإشارة إلى المهام عن طريق الـ Wiki من الوثائق التي تم إنشاؤها بواسطة هذه الأدوات. وإذا لم يتم ذلك، يتم إنشاء جميع الوثائق عن طريق أدوات معينة وإنشاء بعض المهام المحددة التي تدعم الوثائق المكتوبة (مثل مخططات النماذج). وقد يتم تصدير هذه المهام (على هيئة صور مثلاً)، ومن ثم يتم تضمينها في الوظائف. أما مساوئ هذه الطريقة فتتمثل في الحفاظ على القدرة على التتبع والتماسك يدوياً. ومن ناحية أخرى، تنشأ هذه الحاجة بشكل متكرر إذا لم تكن الأدوات المختلفة المستعملة متكاملة مع بعضها بعضاً.

إن استخدام التطبيقات المرتبطة بالإنترنت يتيح مركزة الأدوات المستخدمة، وتبقى هناك إمكانية لتوفير قدرة عالية للوصول إلى المعلومات. لهذه الخاصية تأثيرات إيجابية على إمكانيات التشاركية التي توفُّرها الأدوات. ونظراً إلى عدم توفُّر قواعد بيانات موزَّعة، تكون عملية المشاركة مبسطة، ويتم عرض التعديلات فوراً على المهتمين من دون الحاجة إلى إجراءات متخصصة ومتزامنة.

تم تطبيق أدوات الـ Wiki بنجاح في دراسة حالة مشروع الأستديو العالمي، ويتم توفيرها عن طريق قرص ضوئي مدمج. وقد كان باستطاعة الفريق المركزي تجهيز مشروع تطوير البرمجيات الموزَّعة المراكز باستخدام أدوات الـ Wiki في معظم عمليات التوثيق بشكل أفضل. أما المهام الرئيسة التي تم إجراؤها عن طريق أدوات الـ Wiki فهي المتطلبات (حالات الاستخدام والنماذج ونماذج الشاشات)

والهيكلية (عروض مختلفة بمستويات مختلفة من التفاصيل) والوثائق المتعلقة بالتصميم وخطة المشروع (الفرق والوظائف والتنظيمات والأدوات والبنية التحتية، وغير ذلك) والبرنامج الزمني ومجالات التعاون. تتيح صفحات المناقشة لكل صفحة Wiki المحافظة على الأسس المنطقية للقرارات التي يتم اتخاذها، كما تتيح خاصية إرسال الإشعارات زيادة الإدراك حول الأمر. تم مؤخراً تفعيل المناقشات المباشرة في منتدى المناقشة. تم دعم عمليات التتبع بواسطة أداة خاصة. كما تم تزويد الفرق بالقوالب الرئيسة لبعض الأنشطة الخاصة بالمنتج. وقد اهتم الفريق المركزي بشكل خاص بعملية صيانة هيكلية المعلومات والقدرة على الوصول إليها، ومن ثم تطبيق أدوات الـ Wiki لتعمل كقاعدة معلومات مركزية. واحتفظ الفريق المركزي بتسجيلات مرئية للاجتماعات المهمة تشمل تسجيلات لمحتويات الشاشات التي تم توفيرها من خلال أدوات الـ Wiki. وقد تم استخدام التسجيلات المرئية للمتطلبات والهيكلية كمرجعية للاجتماعات. على سبيل المثال، تم تزويد الفرق بها لفهم المتطلبات ومنطقية التصميم وسياق العمل بصورة أفضل. واستُخدمت تسجيلات محتويات الشاشة كبديل للدليل المكتوب عن عمليات التنصيب والتطبيق. ويمكن استخدام تسجيلات محتويات الشاشة المدعومة بالصوت لتخيُّل المهام التقنية بصورة أفضل؛ على سبيل المثال، عندما تمت مناقشة اللغة الموحدة لتشكيل النماذج شفوياً.

تختلف الحاجة للاتصالات والتوثيق وإدارة المعرفة من مشروع لآخر وفقاً لطبيعة المشروع. سوف تختلف طبيعة الوثائق، فقد يتم إنشاؤها بصورة رسمية أو عادية وفق المنهجية المستخدمة. عادةً ما تكون قاعدة بيانات المتطلبات رسميَّةً في المشاريع الكبيرة؛ كما تكون أكثر تنظيماً وتكون عمليات إدارة الإعدادات أكثر دقة. لا يوجد

توصيات محددة حول البنية التحتية لإدارة المعرفة لأنها تعتمد بشكل كبير على متطلبات المشروع والخيارات المتاحة. بشكل عام، يجب أن تكون الأدوات البسيطة مفضلة على الأدوات التي تتطلب عمليات محددة وهيكلية خاصة لتوفّر المرونة وسهولة الاستخدام.

12-4 إدارة إعداد البرمجيات

تعرف إدارة إعداد البرمجيات بأنها إدارة الطريقة التي يتم بها إنشاء البرمجيات وتعديلها من خلال نظام التحكم بالشيفرة البرمجية المصدرية (Source Code)، والتحكم بالمراجعة، وتتبع إنشاء الكائنات البرمجية وبناء الإصدار البرمجي عندما تصبح جاهزة. ويتضمن ذلك تحديد إعدادات البرمجية عند نقطة زمنية معينة والتحكم المنظم بالتغييرات التي تطرأ على الإعدادات وضمان أن تكون الإعدادات قابلة للتتبع بشكل مستمر، وأن لا تواجه أي مشكلة في عمليات التوافق خلال فترة عمل المشروع (Paulk [et al.], 1993).

تلعب إدارة إعداد البرمجيات دوراً محورياً في نجاح أي مشروع برمجي. تشير خبرتنا أنه عند زيادة الضغط على مشروع التطوير البرمجي، يتم التخلي عادة عن العمليات، خصوصاً تلك التي تُنَفَّذُ في الشركات المبتدئة قليلة الخبرة، وأول ما يتم التخلي عنه العملية التي تتحكم في إدارة واستخدام البنية التحتية لإدارة إعداد البرمجيات.

إتسع نطاق الإعدادات البرمجية عبر السنين لتشمل إدارة إعداد المهام وليس فقط البرمجيات. ولذلك، يجب أن تتم إدارة هذا النشاط بفاعلية منذ بداية مشروع التطوير البرمجي ولا ينتهي بانتهائه، ولكن يستمر حتى انتهاء جميع أجزاء البرمجية.

12-4-1 اختيار البنية التحتية لإدارة إعداد البرمجيات

في إدارة إعداد البرمجيات في مشاريع تطوير البرمجيات الموزَّعة المراكز، من المهم أن يستلم أي عضو من أعضاء الفريق جميع المعلومات المتعلقة بالعمل الذي يقوم به أعضاء الفريق الآخرون وكيفية سير العمل في المشروع والوضع الحالي للمشروع والتغييرات التي تم تنفيذها والعضو الذي قام بها، وغير ذلك. عند التعاون في برمجة نفس الشيفرة البرمجية المصدرية في العمليات الموزَّعة المراكز، يجب ضمان وجود آلية نقل مستمرة ومرنة (عند الطلب) لأن المبرمجين يحتاجون إلى الإطلاع على التغييرات التي يجريها كل منهم (Asklund [et al.], 1999).

من المهم أيضاً دعم القدرة على مشاركة الملفات والتغييرات المتزامنة. أما الحلول التي تتضمن استخدام خاصية «الإغلاق» والسماح بدخول الأشخاص المصرَّح لهم إلى الملفات فيجب أن تعمل بفاعلية وكفاءة نظراً إلى صعوبة التعامل مع الوضع عندما يكون أعضاء الفريق موجودين في مواقع مختلفة ويضطرون إلى انتظار بعضهم بعضاً. ويجب أن يتم كل هذا بالطبع من خلال شبكات متعددة (وغير منفصلة) من أجل دمج العمل في المواقع المختلفة.

12-4-2 البنية التحتية لإدارة إعداد البرمجيات

في مشروع الأستديو العالمي، استُخدمت في البداية أداة لإدارة إعداد البرمجيات لم تكن مناسبة للتطوير الموزع من خلال الشبكات العامة والشبكات متوسطة السرعة؛ وقد كان ذلك متوقعاً وفقاً لخبرتنا، وأيضاً وفقاً للمعلومات الموثقة حول هذا الموضوع. وحين الانتقال إلى مرحلة التطبيق المتقدمة في المشروع حيث يعمل على المشروع عدة فرق موزَّعة، أدرك الفريق المركزي أهمية البنية التحتية لإدارة إعداد البرمجيات، وقرروا الانتقال إلى نظام تحكم معدّل يتيح

القدرة على الوصول إلى المعلومات باستخدام بروتوكول نقل النص التشعبي الاعتيادي، وبالتالي اكتساب القدرة على الدخول حتى في نظام الشبكات المقيدة. يتم استخدام برنامج خاص لربط البرمجة بنظام المصدر المفتوح لتحقيق التكامل بين لغة الـ visual studio (نظام التطوير المدمج الذي تم اختياره بشكل مبدئي) والبرنامج المعدل الذي يتم استخدامه. ولكن التجارب أظهرت أن ذلك غير قابل للتطبيق خلال عمليات الإنتاج. وهناك ثمة أمور أخرى تتعلق بالاستخدام المزدوج لنظام التحكم الذي تم تعديله ولغة الـ visual studio، لكن الفريق المركزي كان قادراً على التغلب عليها.

12-4-2-1 إدارة التكامل والوحدات المبرمجة

تُعتبر عملية تنسيق تطوير البرمجيات الموزَّعة المراكز من عوامل النجاح المهمة. وتصبح عمليات إنشاء الوحدات المبرمجة[*] أوتوماتيكياً والدمج المستمر، التي تتم كل ليلة، جزءاً مهماً في تنظيمات البرمجيات التعاونية وإدارتها. وفي بيئة العمل التي يوجد فيها عدد من الأشخاص وعدة قواعد عمل، يتم توفير آراء وانطباعات وملاحظات فورية للمبرمجين في ما يتعلق بقبول الشيفرة البرمجية الجديدة، كما يقومون بتسهيل جمع العمل المنجز للأجزاء المختلفة أو المتماثلة من المشروع. إن إنشاء الوحدات المبرمجة يومياً بالاعتماد على النسخة البرمجية الأخيرة كل مرة من شأنها تعويض الإخفاقات التقنية وتختبر البرمجيات الجديدة وتحدد المشاكل المحتملة وتعرض النتائج بشكل فوري على المبرمجين الموزعين في مؤسسات ودول مختلفة (Undrus, 2003).

(*) الوحدة المبرمجة هي جزء من البرمجية قابل للتركيب والعمل بما تتضمنه من مزايا وخصائص. وتُستخدم لتركيب البرمجية لأغراض الفحص والاختبار غالباً. ويمكن الحصول على هذه الوحدات البرمجية أوتوماتيكياً باستخدام أدوات معينة كأداة apache ant.

هناك تشكيلة واسعة من الأدوات ذات المصادر المفتوحة التي يمكن استخدامها في عمليات البرمجة والدمج. ومن أمثلتها أداة إعداد الوحدات المبرمجة (NAnt)، والأداة (NUnit) التي تُستخدم في اختبار الوحدات البرمجية وعمليات الدمج. أما بالنسبة إلى الأداة (.NET)، فيقوم البرنامج (CruiseControl.NET) بدمج (NAnt) و(NUnit) والنظام الذي تم تعديله معاً، وبالتالي يمكن استخدام تلك الأدوات بفاعلية لعمليات الدمج المستمرة (Fowler and Foemmel, 2005)، إذ إنها مفيدة في الاختبار وإنشاء الوحدات المبرمجة أوتوماتيكياً. وهذا يعني أن المبرمج يقوم في كل مرة بتسليم الشيفرة البرمجية التي كتبها وتضمينها في النظام. وتقوم الأداة باختبار تلك الشيفرة البرمجية على النسخة الأخيرة من النظام. يتم ترجمة نتائج هذه البرمجة والاختبارات المتعلقة بها على شكل سجلات لوصف الحالات وتسجيل الأخطاء، ويمكن الوصول إلى هذه السجلات بسهولة باستخدام تطبيقات بسيطة من خلال الإنترنت تكون بمثابة نافذة للدخول إلى المشروع. بالتالي، يتمكن جميع المبرمجون من الحصول على بيانات الوضع الحالي للدمج والاختبارات والوصول إليها بسهولة. ويمكن الحصول على ميزة أخرى باستخدام أداة (NAnt) وهي عملية الإنشاء التلقائي للوثائق المتعلقة ببنّية برمجة التطبيقات (API) التي يتم توفيرها في الموقع الإلكتروني أيضاً، بحيث يكون الوصول إليها سهلاً.

12-4-3 إدارة إعداد البرمجيات لتسهيل التطوير البرمجي الشامل

لن تحقق البنية التحتية لإدارة التهيئة (CM) أهدافها من دون اقترانها بعملية مُعَدَّة ومنفذة بشكل جيد. وتشمل عملية دعم إدارة التهيئة ما يأتي: (1) تحديد «رسمي» للأمور التي تطبق عليها (نموذج

العملية)، و(2) الآلية التي يتم تأكيد العملية بشكل فعلي من خلالها (Estublier, 2000). وقد لوحظ أيضاً أن دعم العملية الأفضل والأكثر مرونةً هي الخاصية التي تفتقدها معظم أدوات إدارة التهيئة، وذلك لأن جميع أنظمة إدارة التهيئة تعتمد على دورة حياة المنتج المعدة مسبقاً وغير المرنة (Estublier, 2000).

نظراً إلى الجهود المبذولة لتوزيع مشروع التطوير البرمجي في المناطق الجغرافية المتعددة، ظهرت الحاجة إلى وجود عمليات برمجة متزامنة في جميع المجالات بما في ذلك مخطط التنفيذ. ويمكن القيام بتغييرات متعددة على نفس المهمة أو على عدة مهام مختلفة قبل توزيع الوحدات الجزئية لأحد الأنظمة البرمجية على المناطق المختلفة وعملها بشكل متزامن.

يمكن للمبرمجين الذين يُطلب منهم العمل في فروع برمجة مختلفة ومتوازية العمل في بيئات برمجية معزولة (sandbox) بغض النظر عن الطبيعة الحالية للعمل، وبذا يتجنبون استخدام تغييرات أخرى مؤقتة. على كل حال، يؤدي العمل بشكل منعزل إلى عدم معرفة المبرمجين بما يقوم به كل منهم، وبالتالي يكون هناك احتمالية أن يقوم بعضهم بإجراء تغييرات لا تتوافق مع عمل بعضهم بعضاً. يجب أن يتم حل هذه المشاكل حين دمج هذه الفروع في فرع واحد للبرمجة.

يمكن القيام بعملية البرمجة المتزامنة بعدة مستويات. على مستوى النظام، يتم تطوير المنتجات الفرعية أو الوظائف المختلفة بشكل متوازٍ مع بعضها (تطوير متزامن للنظام)؛ وفي مستوى أدنى أكثر تفصيلاً، يمكن أن يقوم عدد من المبرمجين بإجراء تغييرات متزامنة على عدة إصدارات لنفس الملف (برمجة متزامنة للبرنامج). كلما كان مستوى البرمجة أقل، كان إجراء عمل متزامن ممكناً بشكل

أكبر. ولكن تكون المخاطر أكبر من وجود عدم توافق في التغييرات. وهناك طريقتان رئيستان لتقليل مخاطر عدم وجود توافق:

أولاً، تؤدي الهيكلية المتماسكة للمنتج (بنية تصميمية) إلى إتاحة جعل الروابط أكثر وضوحاً وتجزيء العمل إلى عدة أقسام يتم العمل عليها. كلما كانت الأجزاء غير مترابطة ولا تعتمد على بعضها بعضاً، كان من الأسهل وصف علاقاتها، وكان احتمال وجود تعارض بينها قليلاً. حينها يكون توزيع المسؤوليات المتعلقة بالأجزاء المختلفة للمنتج على الأفراد المبرمجين أو على مجموعات العمل في المشروع أمراً سهلاً نسبياً.

ثانياً، إن وجود مستوى مرتفع من الإدراك للعمل الذي تم إنجازه والعمل قيد الإنجاز حالياً من قبل المبرمجين الآخرين أو مجموعات المشروع الأخرى يساعد في تقليل مخاطر وجود تعارض. إن البرمجة المتزامنة التي تتم في مستويات بسيطة أكثر تفصيلاً تتطلب إدراكاً أكبر من البرمجة التي تتم في مراحل متقدمة. في العمل المتزامن على مستوى النظام، قد تكون معرفة إذا ما تم تغيير العلاقات وفي أي وقت تم تغييرها أمراً كافياً؛ لكن عندما يعمل مبرمجان اثنان على نفس الوحدة، فإنهما يحتاجان على الأرجح إلى معرفة إذا ما كان الطرف الآخر قد بدأ العمل في نفس الملف ومتى بدأ ذلك.

12-4-3-1 المهام المحددة بدقة

تنشأ المهام المحددة بدقة نتيجة الإدراك المتزايد بأن هناك من يعلم بما يقوم به الآخرون. وهي من الأمور المهمة التي تتعلق بالتعاون الذي يتم بين فرق العمل عبر مسافات كبيرة. كما إنها تقلل خطر حصول سوء فهم (نتيجة اختلافات ثقافية على سبيل المثال).

لكن ذلك لا يحقق إدراكاً حقيقياً في دعم النظام، أي لا يوضح الأحداث التي حدثت في الماضي أو تلك التي تحدث الآن. فعن طريق تحديد مهام كل عضو من أعضاء فريق التطوير لن يقوم أي منهم بالتقليل من أهمية العمل المتزامن عند مستويات العمل المنخفضة.

12-4-3-2 المسؤوليات الحصرية

يُعتبر تحديد المسؤوليات الحصرية (على سبيل المثال، مجموعة من الملفات أو الوحدات) من التقنيات الشائعة الأخرى. خصوصاً عند وجود مشروع برمجي جديد، وهذا يكون ممكناً عن طريق إنشاء هيكلية مناسبة للمنتج، وتقسيم العمل بين المبرمجين بحيث يصبح كل منهم مسؤولاً عن أجزاء مختلفة من المنتج. وينتج من وجود مهام واضحة وعلاقات محددة عمل جيد، خصوصاً بين المواقع التي تعمل بشكل منفصل. وعلى الرغم من ذلك، فإن الإدراك من الأمور المهمة، فلذا تجد بعض مشاريع تطوير البرمجيات الموزَّعة المراكز تبحث عن روابط أفضل بين المبرمجين في المواقع المختلفة تتيح إمكانية تبادل العمل يومياً وليس خلال الاجتماعات الأسبوعية فقط. وذلك لأنه من الممكن أن يقوم شخص واحد فقط بإجراء تغييرات على أحد الملفات كنتيجة لتقسيم المسؤوليات، ومن النادر استخدام الفروع الجزئية على مستوى الملفات.

من الصعب إجراء تقسيم للمسؤوليات بنفس الطريقة خلال عمليات الصيانة. فبدلاً من تجزئة المهام المتعلقة بالصيانة إلى مهام صغيرة جداً، يجب أن ينفذها أشخاص مختلفون وفق مجالات مسؤولياتهم؛ من الأفضل أن نمكن المبرمج الذي لاحظ وجود خطأ بسيط في وحدة العمل لدى مبرمج آخر (قد تكون الوحدة ضمن فروع عمله) أن يقوم بتعديل فوري لهذا الخطأ، ولو مؤقتاً، ويقوم

بعدها باختبار التغييرات التي قام بها. وهذا أمر مهم جداً خصوصاً عندما يوجد المبرمجون في مواقع مختلفة (على الرغم من أن الوجود في المواقع المختلفة يصعب من تنفيذ ذلك). يجب حينها إبلاغ المسؤول عن الوحدة أن شخصاً آخر قام بإجراء تعديل مقترح على وحدته، ليقرر عندئذ إذا ما كان يجب دمج هذا التعديل في الفرع الرئيس للعمل.

إذا كان عدد أجزاء المسؤوليات التي تم تقسيمها أكبر من اللازم، تصبح مسؤولية كل مبرمج محدودة جداً بحيث يكون المبرمجون مقيدين في مجال العمل. يعتقد بعضهم أن أسلوب العمل الخاص بمشروع التطوير الجديد الذي نتحدث عنه ينتج فقط من عدم وجود دعم كافٍ لعمليات التطوير الموزَّعة، وهذا من الأسباب التي تدعو إلى الحصول على إدارة التهيئة أكثر بساطة من دون وجود فروع عمل. بدلاً من ذلك، يجب أن يُتبع أسلوب العمل الطبيعي بحيث يطبِّق كل وظيفة جديدة شخص مسؤول، وهو الشخص نفسه الذي يقوم بتطبيق الملف الذي تم إجراء التغييرات عليه كاملاً. ومن الواجب أيضاً القول إن الصيانة وعمليات البرمجة الإضافية لمنتج ناجح هي من أهم المراحل التي يمر بها المشروع (يشكل ذلك وفقاً لبعض المصادر حوالى ثمانين بالمئة من المشروع). لذلك، وعلى الرغم من أن الاهتمام بإجراء عمليات الإنشاء للأدوات والعمليات لمشروع برمجي جديد أمر شائع أكثر من الاهتمام بالصيانة وعمليات البرمجة الإضافية، غير أن ذلك يُعتبر خطأً كبيراً وفقاً لما تم بحثه.

فكرة مفيدة: بعض القواعد الإجرائية لإدارة إعداد البرمجيات (Asklund [et al.], 1999):

1. كرر المعلومات في مواقع البرمجة المختلفة بحيث

يعمل كل فرد على خادم محلي مستقل. وتأكد دائماً من أن المعلومات متزامنة وحديثة، ويفضل أداء ذلك تلقائياً باستخدام أداة تدعم ذلك.

2. لا تضع حماية على الملفات لمنع إجراء عمليات تطوير متزامنة، خصوصاً إذا كانت الحماية تقيّد عمل المبرمجين في مواقع مختلفة. بدلاً من ذلك، اجعل المبرمجين قادرين على إنشاء فروع مؤقتة خاصة بهم. يجب دمج الفروع المؤقتة في ما بعد مع الفروع الأصلية في أقرب زمن ممكن والتخلص من الفروع المؤقتة.

3. قم بإنشاء بنية تصميمية جيدة للمنتج بحيث ينتج منها تقسيم طبيعي للعمل. تقلل الهيكلية الجيدة الحاجة إلى الفروع. وإذا تم الاستمرار في استخدام الفروع، تقلل الهيكلية الجيدة من مخاطر عدم التوافق عند إجراء أي عمليات دمج محتملة.

4. لا تقم بتجزيء العمل إلى جزئيات صغيرة جداً لتوزيعها على المبرمجين المسؤولين عن ملفات أو وحدات محددة، إذ إن ذلك يؤدي إلى الجمود وعدم المرونة في التعامل مع التغييرات، ولا يمكن التعامل مع عدة تغييرات تؤثر في الملف نفسه بشكل متزامن، بالتحديد إذا كانت التغييرات المختلفة قد تؤثر في المسؤوليات في المناطق المختلفة.

12-5 الملخص والاستنتاجات

إن الركائز الثلاث لإعدادات بنية تحتية فعالة في مشاريع التطوير الموزَّعة المراكز هي التواصل والتعاون، وإدارة المعرفة، وإدارة إعداد البرمجيات. وللحصول على بنية تحتية تتيح التواصل والتعاون

الفعال، يجب التركيز على القدرة على الوصول إلى المعلومات من قِبَل جميع المهتمين بالمشروع وعلى إنشاء بيئة تشاركية بحيث يساهم جميع الأفراد في المشاركة في المعلومات. إن مبدأ وجود قاعدة معرفة هو محور البنية التحتية لإدارة المعرفة. في الحقيقة، يجب ألا تتكون قاعدة المعرفة من أداة واحدة بل قد تتكون من مجموعة من الأدوات المتكاملة جيداً (Gersmann, 2005). طالما أن إدارة إعدادات البرمجية أمر مهم، فقد لاحظ إستبلييه (Estublier, 2000) أن الدعم الأفضل والأكثر مرونة للعملية هو من أكثر الخصائص التي تفتقدها أدوات إدارة التهيئة، لأن معظم هذه الأدوات تعتمد على إعداداتها، وبالتالي تفتقد إلى المرونة خلال تعاملها مع المنتج.

وهذا ما يجعل إدارة إعداد البرمجيات في مشاريع تطوير البرمجيات الموزَّعة المراكز أكثر تحدياً. وكما إن هدف التقنية توفير احتياجات الأعمال التجارية، فلن تحقق البنية التحتية لإدارة التهيئة أهدافها دون تضمينها مع عملية مخَطَّطٍ لها بشكل جيد ومنفَّذة أيضاً بشكل جيد. ويعدُّ الحد من اعتماد المبرمجين على بعضهم بعضاً من الاستراتيجيات الناجحة في أي عملية برمجة، خصوصاً إذا كانوا يوجدون في أكثر من موقع. ويمكن تنفيذ هذا بشكل رئيس خلال عملية إنشاء هيكلية (الهيكلية) المنتج المراد تطويره. ويتم تقسيم النظام إلى وحدات أو أجزاء، ومن ثم تبرمجها مجموعات مختلفة وبشكل منفصل. على كل حال، اتضح أنه وعلى الرغم من وجود هيكلية جيدة، غير أن العلاقات تدوم بين المكوّنات. يتضح هذا حين ظهور الحاجة إلى تعديل الروابط أو حين الحاجة إلى دمج المكوّنات. هذه بعض الأمثلة على حالات يحتاج فيها المرء (على الرغم من أنه حاول تجنب ذلك) إلى الإطلاع على عمل مجموعة تعمل على مكوّنات مختلفة وإيجاد تزامن بين أعمالهم (Asklund [et al.], 1999).

12–6 أسئلة للمناقشة

1. ما هي الركائز الثلاث لإعداد بنية تحتية في مشاريع التطوير الموزَّعة؟
2. متى يجب إنشاء البنية التحتية لمشاريع التطوير الموزَّعة؟ وما هي الفترة التي يجب أن تستمر فيها؟
3. في مجال مشاريع التطوير الموزَّعة، هل تدعم أداة مخططات التنفيذ (CM) سير العمل بشكل أفضل أم الأداة التي تساعد على تحديد وصيانة عملية مرنة؟ ولماذا؟

المراجع

Books

Asklund, U., B. Magnusson, and A. Persson. *«Experiences: Distributed Software Development and Configuration Management.»* Lund, Sweden: Lund University, 1999.

Tellioglu, Hilda and Ina Wagner. *«Negotiating Boundaries - Configuration Management in Software Development Teams.»* Vienna, Austria: Vienna University of Technology, 1995.

Periodicals

Herbsleb, James D. and Deependra Moitra."Global Software Development.» *IEEE Software*: vol. 18, no. 2, 16-20 March/ April 2001.

Paulk, Mark [et al.]. «Capability Maturity Model for Software, Version 1. 1.» CMU/ SEI-93-TR-024, 1993.

Perry, D. E., N. A. Staudenmayer, and L. G. Votta. «People, Organizations, and Process Improvement.» *IEEE Software*: vol. 11, no. 4, July/ August 1994, pp. 36-45.

Undrus, Alexander. «NICOS System of Nightly Builds for Distributed Development.» *CHEP 2003*, San Diego, CA, 2003.

Conferences

Estublier, Jacky. «Software Configuration Management: A Roadmap.» *Proceedings of the Conference on The Future of Software Engineering*, Limerick, Ireland, 2000, pp. 279-299.

Thesis

Gersmann, S. «Development of Strategies for Global Software Development.» Technische Universität München (Munich, Germany) and Siemens Corporate Research (Princeton, N.J.), Masters Thesis, 2005.

Websites

Fowler, Martin and Matthew Foemmel. «Continuous Integration.» Retrieved 09/27/2005, from: <http://www.martinfowler.com/articles/continuousIntegration.html.>.

الفصل الثالث عشر

التواصل

كانت شركة متعدِّدَة الجنسيات (MNC) تعمل في مشروع تطوير برمجيات موزَّعة منذ ستة شهور عندما اكتشفت أن شبكة معقَّدَةً جداً من عمليات المشروع قد سيطرت على ستة مواقع مختلفة للشركة حول العالم. جاء بعض المهندسين في بعض المواقع للتعرف على نظرائهم في مواقع أخرى، ما أدى إلى ظهور عدد لا يحصى من قنوات التواصل بينهم. في العادة، يقوم هؤلاء المهندسون بإيصال احتياجاتهم، ويتفقون على ما يتم تسليمه مؤقتاً، وأنشأوا وتبادلوا المنتجات البرمجية التي تقوم بتنفيذ تلك المتطلبات. ومن ناحية أخرى، كان هناك مهندسون في بعض المواقع معزولون، ونادراً ما يتواصلون مع نظرائهم الذين يعملون عن بُعد، ما أدى إلى ضعف فهم العمل المخصص لهم وعدم تناسقه واتصافه بالغموض.

تقع على عاتق شركة (MNC) الآن مهمة شاقة وهي تنظيم استخدام قنوات التواصل. فحين ترك التواصل من دون تحكُّم أو سيطرة، سيكون هناك عدد كبير جداً من حالات التواصل ما يؤدي

إلى وجود ضغط كبير غير مبرر وشبكة معقدة من العمليات قيد العمل؛ ويؤدي وجود عدد قليل جداً من قنوات التواصل إلى نقص في تبادل المعلومات وضعف المشاركة في المعرفة وعمل الفريق. فما هي المعادلة الصحيحة لذلك؟ كيف يمكن أن تحل الشركة متعددة الجنسيات هذه المشكلة؟

سوف نسلِّط في هذا الفصل الضوء على أدوات التحكم والعوائق المفروضة على التواصل في مشاريع تطوير البرمجيات الموزَّعة المراكز. وسيؤدي فهم ذلك إلى استيعاب أكبر لاستراتيجيات التطوير اللازمة للتعامل الفعَّال مع التواصل بين الفرق الموزَّعة وتنظيم فعال للأنشطة التي يقومون بها أيضاً. وقد تم أيضاً مناقشة التقنية التي يمكن استخدامها لتتبع التواصل بين الفرق الموزَّعة في عدة مواقع لإظهار العلاقات بين روابط الهيكلية للمشروع. تتطلب الروابط المهمة والحرجة اتصالات مكثفة، وهذه التقنية تساعد المديرين في تحديد أنواع الروابط المهمة وتعزيز التواصل المكثف بين الفرق المتعاونة.

13-1 أدوات التحكم بالتواصل

تمثِّل علاقات الترابط بين المهام ووجود روابط في التنظيم ـ وفقاً لخبرتنا ووفقاً للنظريات الموجودة ـ عاملاً محفزاً مهماً لنقل المعلومات بين الفرق المتعاونة (Sosa [et al.], 2002). كلما كان مستوى علاقات الترابط كبيراً بين المهام، كانت الحاجة للاتصالات أكبر؛ وتزداد هذه الحاجة بشكل أكبر إذا تخلل هذا التواصل شعور بعدم التأكد والغموض. في هذه الظروف، تتفاعل الفرق المتعاونة من أجل:

- الحد من الغموض: لمواجهة المعلومات غير الدقيقة والحد من تأثيرها، يجب الحد من الغموض وتحويله إلى مجموعة من المسائل المحددة جيداً أو الوصول إلى اتفاق لحلها.
- الحصول على الحد الأعلى من الثبات: عند وجود نقص في المعلومات، يجب تبادل المعلومات المهمة والحرجة (حالما تكون متوفّرة) وذلك لمعالجة هذا العيب.

إن وجود روابط في التنظيم تجعل الأعضاء المنتمين إلى نفس الفريق يشعرون بهوية واحدة تربط بينهم. كما تؤدي هذه الروابط إلى وجود اتصالات مستمرة بشكل أكبر في ما بينهم لشعورهم بالاختلاف عن أعضاء الفرق الأخرى.

تحقق أدوات التحكم هذه هدفاً أكبر في مشاريع تطوير البرمجيات الموزَّعة المراكز، إذ إن تنفيذها من قبل أعضاء ينتمون إلى منظمات موزَّعة في مناطق جغرافية مختلفة يمثل تحدياً لعمليات التنسيق والتحكم.

فكرة مفيدة:

قم بالحد من روابط المهام.

يجب أن يحدد المسؤولون المهام ذات العلاقات المتبادلة، ويعملوا على تقليل الروابط بين المهام التي تنفذها الفرق الموزَّعة في مناطق جغرافية مختلفة والتي لا ترتبط بروابط تنظيمية مع بعضها (على سبيل المثال، لم يعملوا مع بعضهم بعضاً مسبقاً). يتطلب تطوير البرمجيات، بخاصة في مراحل التخطيط، إجراء اتصالات مكثفة. وسوف تؤدي المسافة الكبيرة بين الفرق الموزَّعة، فضلاً عن عدم معرفتهم لبعضهم بعضاً، إلى أن تكون عملية التواصل صعبةً جداً بينهم.

فكرة مفيدة:
قم بتسهيل التواصل بين الفرق التي تعمل على مهام مترابطة.
عند وجود روابط حرجة للمهمة، يجب أن يقوم المسؤولون بتسهيل التواصل المكثف بين الفرق المضطلعة بالعمل على هذه المهام. وقد يتطلب التغلب على المسافة الكبيرة بين الفرق وعدم معرفة أعضاء الفريق بعضهم بعضاً إيجاد روابط قوية في ما بينهم.

13-2 عوائق التواصل

لقد وجدنا أن العوامل الآتية تعيق تبادل المعلومات بين فرق البرمجة عندما تتواصل مع بعضها (Sosa [et al.], 2002):

- **بعد المسافة**: تؤثر المسافة سلبياً في التواصل. يكون عامل المسافة مهماً عندما تتجاوز خمسين متراً؛ فعندما تزيد المسافة على ذلك يصبح هناك انهيار حاد في التواصل، ولن يكون هناك فرق إذا ما كانت الفرق المتعاونة تقع في مبنيين مختلفين أو مدينتين أو دولتين أو حتى قارتين (Allen, 1984).

- **تداخل أوقات العمل**: تقل احتمالية وجود اتصالات متزامنة بين الفرق المترابطة عندما تقلّ الفترات التي يتداخل فيها العمل بين المواقع المختلفة. ويكون احتمال التواصل وجهاً لوجه منخفضاً جداً؛ يتم استخدام الهاتف للتواصل بين الفرق التي يوجد بينها فترات تداخل في أوقات العمل، والبريد الإلكتروني هو الطريقة المفضلة في حال عدم وجود أي تداخل في أوقات العمل.

فكرة مفيدة:

إذا كان ممكناً، تجنب المواقع التي تعمل عن بُعد ذات الفارق الكبير في التوقيت عن الموقع المركزي.

الصعوبة التي تواجهها المواقع المركزية في أميركا الشمالية في عملها مع مواقع تعمل عن بُعد في الهند والصين هو وجود فارق كبير في التوقيت بينهما. وهذا يقلل من إمكانية وجود اتصالات هاتفية عفوية. حتى عندما يتم وضع مواعيد زمنية للمؤتمرات كالاجتماع التحضيري اليومي، فمن المرجَّح أن يكون على الأقل شخص واحد من المشاركين مرهقاً. تعرفنا على مدير تزويد مقره في شرق الولايات المتحدة يقوم بالتواصل مع الفريق الذي يعمل لديه الموجود في الهند قبل أن يخلد إلى النوم ليلاً. ويقوم بالتواصل بهم مجدداً بعد استيقاظه، وذلك لأنهم كانوا يعملون بينما كان هو نائماً. وحين ظهور مشاكل في العمل، يتسبب ذلك في إقلاق راحته وعدم نومه جيداً. وقد يكون تجنب هذه الفروقات الكبيرة في التوقيت أمراً غير ممكن لأسباب تتعلق في العمل التجاري والتسويقي. فعندما تواجه وضعاً مماثلاً، حاول تخصيص مجموعات عمل للموقع الذي يعمل عن بُعد لا تتطلب إجراء اتصالات هاتفية، أو تحرير بإعداد بنية تنظيمية، بحيث يتم تحرير مع إحدى المواقع الأخرى بواسطة الأقمار الصناعية (على سبيل المثال، الربط مع إحدى المواقع الأوروبية التي ستتقاطع أوقات العمل فيها مع كل من المنطقة الشرقية في الولايات المتحدة ومع الهند أيضاً).

- **الاختلافات الثقافية واللغوية**: ثمة نزعة طبيعية لدى البشر تتمثل في البحث عن المشابهين لهم للتواصل معهم. لذلك، تكون فرص التواصل أكبر بين فريقي عمل إذا كانت الفروق الثقافية واللغوية ما بينهم قليلة. أما وسائل التواصل المفضلة عند وجود اختلافات ثقافية ولغوية كبيرة بين الفرق المتعاونة فهي الوسائل الكتابية كالبريد الإلكتروني.

فكرة مفيدة:

قم بتسهيل وسائل التواصل الكتابية غير المتزامنة.

يجب أن يقوم المسؤولون بتسهيل وسائل التواصل الكتابية غير المتزامنة بين الفرق المتعاونة التي يكون بينها اختلاف كبير في اللغة، وهذا يساعد أعضاء الفريق على تنمية العلاقات التواصلية التي تلعب دوراً مهماً في حصول الشخص على المعلومات وتعلّم كيفية أداء العمل وإيجاد حلول للمهام الذهنية المعقدة. وقد أظهرت الأبحاث أن الوثائق ذات قيمة أقل في الحصول على المعلومات، في حين يحصل مهندسو البرمجيات على معلوماتهم من الآخرين (Kraut and Streeter, 1995). لذا، ما من بديل يغني عن العلاقات الشخصية.

- **الدافع التنظيمي والثقة**: في مشاريع تطوير البرمجيات الموزَّعة المراكز التي تشتمل على منتجات ستحل مكان الأنظمة القديمة، وتكون التنظيمات التي تدعم الأنظمة القديمة أقل ثقة بالشركاء على الأرجح، وذلك لأنهم ينظرون إلى الأمر كتهديد لأمنهم الوظيفي. ومن المرجح أيضاً أن تكون هذه التنظيمات أقل دافعية وحماسة للمشاركة في المشاريع الجديدة.

- **العلاقات الشخصية**: هناك احتمالية ضعيفة للتواصل بين أعضاء الفرق المتعاونة ما لم يتعرف أعضاء هذه الفرق على بعضهم بعضاً.

في مشاريع تطوير البرمجيات الموزَّعة المراكز، وبوجود فوارق في التوقيت بين الفرق، ناهيك بوجود اختلاف كبير في الثقافة واللغة، تكون العمليات الإدارية التي تجرى على هذه الفرق صعبة من ناحية التنظيم والتحكم.

فكرة مفيدة:

كن على دراية بعادات العمل الموسمية في مواقع البرمجة.

سوف تكون فترات العمل في المواقع الموجودة في أميركا الشمالية، في الأغلب، مشابهة لفترات العمل في أميركا الجنوبية. ولكن، قد تكون المواسم معكوسةً. ما يعني عدم توافق فترات العطلات بين الفرق، أو الفترات التي تكون فيها إنتاجية العمل مرتفعة أو منخفضة. ضع خطة برنامج العمل الزمني وفقاً لذلك. ليس من المرجح أن يكون هناك إنتاجية عالية في آخر أسبوع من شهر كانون الأول/ ديسمبر في معظم بلاد العالم.

13-3 التواصل والتنسيق

من المعروف أن التواصل يلعب دوراً مهماً في تحسين أداء تطوير المنتج (Sosa [et al.], 2002)، وتحديداً على تحسين أداء الفرق المتعاونة، وبخاصة عندما تكون المهام مترابطةً (Kraut and Streeter, 1995; Malone and Crowston, 1994). ومع ذلك، وكما تم ذكره سابقاً، يمكن أن يكون تنسيق الأنشطة المترابطة التي تتطلب عملية

اتصالات مكثفة صعباً في مشاريع تطوير البرمجيات الموزَّعة المراكز. يمكن تحسين التواصل والتنسيق عن طريق تطبيق المبادئ الثلاثة الآتية والمستمدة من النظرية التنظيمية (McChesney and Gallagher, 2004).

1. **نظرية التنسيق**: نصَّت النظرية على أن التنسيق هو عملية لإدارة العلاقات الترابطية بين الأنشطة. يمكن استخدام مبادئ نظرية التنسيق لتحديد علاقات الترابط التنسيقية وآليات التنسيق اللازمة لإدارة هذه العلاقات. وفي ما يأتي بعض الأمثلة على علاقات الترابط التنسيقية وآليات التنسيق:

أ) توابع المصادر المشتركة، مثل وحدة البرمجة، يمكن التعامل معها من خلال أداة إدارة التهيئة.

ب) روابط قيود المتطلبات السابقة، كأن يتم الموافقة على طلب إجراء تغيير قبل أن يتم توكيل أحد مهندسي البرمجيات بإجرائه، ويمكن التعامل معه عن طريق استخدام الإجراءات المتعلقة بإدارة التغيير.

ج) روابط إمكانية الوصول، كإيصال معلومات عن إجراء تغيير موافق عليه إلى أحد مهندسي البرمجيات، ويمكن التعامل معه عن طريق نظام تحديد تلقائي أو يدوي للمهام.

د) روابط إمكانية الاستخدام، كالحاجة إلى المواصفات الكاملة لطلب التغيير، ويمكن التعامل معه عن طريق النموذج الموحد لطلبات التغيير ووضعه حيز التنفيذ آلياً.

هـ) روابط ذات قيود مطابقة، كإنشاء الوحدات المبرمجة يومياً، ويمكن التعامل معه عن طريق استخدام إجراء الوحدات المبرمجة المعياري وعن طريق إدارة إعداد البرمجيات.

و) روابط المهام والمهام الفرعية، كتجزئة مهام المشروع إلى مهام جزئية لتسهيل إدارة المهام المعقدة، ويمكن التعامل معها عن طريق هيكلية عمليات تجزئة العمل واستخدام تقنيات التخطيط كـ PERT(*) وCPM(**).

2. **أساليب التواصل**: وهي أساليب يستخدمها عادةً أحد أعضاء مجتمع ما للوصول إلى بعض الأهداف. في مجتمع البرمجيات، على سبيل المثال، تُستخدم المقابلات والاجتماعات في مراحل مختلفة خلال فترة عملية التطوير البرمجي وذلك للموافقة على المهام قيد البحث أو إبداء الرأي حولها.

3. **العقل الجماعي**: إن وجود فهم مشترك من الأمور الأساسية لنجاح أي مشروع، لأنها تعكس مدى فهم الفرق المتعاونة للمسألة التي تقوم بمحاولة حلها. السؤال الذي يتبادر للذهن في هذا السياق، «هل يركزون في الاتجاه الصحيح للوصول إلى الهدف العام؟». وتنص نظرية العقل الجماعي على أنه يتم تعزيز الفهم المشترك من خلال التفاعل بين أعضاء الفرق التعاونية، ومن الصعب الوصول إلى هذا في بيئة العمل الموزَّعة جغرافياً على مناطق مختلفة.

إن آليات التنسيق المستقاة من نظرية التنسيق وأساليب التواصل هي الآليات الرسمية للتواصل، بينما يمكن إيجاد العقل الجماعي من خلال تبادل المعلومات. ووفقاً لخبرتنا، ووفقاً للنظريات القائمة، فإن آليات التواصل الرسمية مفضَّلةٌ في المهام الروتينية وتكون أكثر فعالية عندما تكون المشاريع محددة وواضحة؛ ويلعب التواصل غير

(*) (PERT) Program Evaluation and Review Technique، وهي تقنية تستخدم لإدارة المشاريع مصممة لتحليل وعرض المهام التي يجب إنجازها لإكمال المشروع، خصوصاً الزمن اللازم لإكمال كل مهمة وتحديد أقل وقت ممكن لإنهاء المشروع كاملاً.

(**) انظر: 5-2-4 تحديد المسارات الحرجة.

الرسمي دوراً مهماً في مواجهة عدم التأكد ووجود نقص في المعلومات (Kraut and Streeter, 1995).

لسوء الحظ، تتعامل معظم عمليات تطوير البرمجيات مع مسائل محددة بشكل سيئ ومع معلومات غير مؤكدة وغير مكتملة وعرضة للتغيير في أي وقت. التحدي في التطوير الموزع، وفقاً لذلك، هو تحديد وسائل ذات فعالية وكفاءة عالية من أجل عمليات التواصل الشخصية وغير الرسمية. هذا النوع من التواصل مهم جداً في المراحل الأولية من عمليات التطوير البرمجي، أي عندما تكون في طور التحليل. من المهم أن يمتلك جميع المشاركين فهماً عاماً لهذه المسألة بحيث يقومون بعمل صحيح باتجاه إيجاد الحلول لها. وما إن يتم إنهاء مرحلتي البدء والتطوير، حتى تصبح المعلومات أكثر ثباتاً وأكثر تأكيداً، وتقل الحاجة إلى اتصالات غير رسمية مكثّفة.

فكرة مفيدة:

كوّن فهماً مشتركاً بين الفرق العاملة في المشروع.

يمكن أن يشارك أعضاء الفرق التي تعمل عن بُعد الفريق المركزي خلال مراحل البدء والتطوير التي يتم أثناءها إجراء عمليات التحليل على المتطلبات وإنشاء الهيكلية.

بعد انتهاء مرحلة التطوير، يتم إعادة بعض أعضاء الفرق التي تعمل عن بُعد إلى مواقعهم الأصلية، ليتم اعتبارهم خبراء في هذه المواقع، ويقوموا بخلق فهم مشترك للمتطلبات وتصور البنية التصميمية للمشروع.

خلال مرحلة التنفيذ، وبينما تقوم الفرق التي تعمل عن بُعد بتنفيذ النظام، يمكن أن يقوم الخبراء في الفريق المركزي بالسفر بشكل منتظم إلى المواقع التي تعمل عن بُعد لمشاركة أعضاء هذه المواقع في استخدام النظام

الذي يقومون ببنائه كما هو. تدعى هذه التقنية أحياناً «استخدم منتجك». عن طريق التدريب على الواقع الحقيقي الذي سوف يُستخدم فيه النظام، وسوف يكوِّن الفريق الذي يعمل عن بُعد إدراكاً ومعرفة أفضل للمسألة التي قاموا بحلها.

إن عملية تبادل الأعضاء بين الموقع المركزي والمواقع التي تعمل عن بُعد مفيدة أيضاً لتفاعل الأعضاء وجهاً لوجه وتساعد على بناء مجموعات من المعارف الشخصية للأعضاء.

13-4 التواصل والتحكم

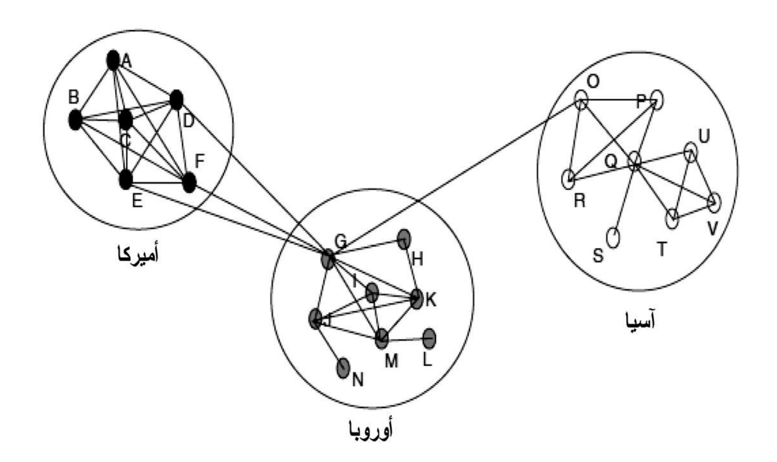
على الرغم من أن التواصل الفعال والتنسيق أمران مهمان جداً لتحسين أداء المنتج، يكون التواصل الزائد ضارّاً. وإذا لم يتم التحكم بالتواصل، سوف يكون هناك قنوات كثيرة من التواصل بين الفرق المتعاونة، ما يؤدي إلى إنشاء شبكة معقدة من العمليات التي تجري ضمن المشروع. يمكن أن يؤدي ذلك أيضاً إلى وجود ضغط كبير على تناقل المعلومات وقد يتم نقل معلومات ليست ذات أهمية، وبالتالي صرف الانتباه عن المعلومات المهمة المتعلقة بالمشروع. فضلاً عن أن التواصل قد يكون مكلفاً مادياً مثل التواصل الدولي وتكلفة الرحلات والتنقلات التي تتم لتحقيق التواصل.

من الاستراتيجيات المتبعة لتعزيز التواصل بأسلوب مسيطر عليه، تقليل روابط المهام الموزَّعة وتسهيل عمليات التفاعل في ما يتعلق بالمهام المترابطة بشكل كبير بين الفرق المتعاونة. وللوصول إلى هذا، يجب امتلاك القدرة على تتبع قنوات التواصل بين الفرق المختلفة للتأكد من أن الفرق التي تعمل على المهام المترابطة تتفاعل

بشكل جيد. ويصف القسم 13-4-1 تقنية يتم اتباعها للقيام بهذا التتبع؛ ويسمى ذلك «تحليل الشبكات الاجتماعية»، إذ بدأت مجتمعات مهندسي البرمجيات بالاهتمام بهذه التقنية.

13-4-1 تحليل الشبكات الاجتماعية

إن عملية تحليل الشبكات الاجتماعية (SNA) هي عملية مبنية على تقنية المسح لإظهار التفاعلات المتبادلة بين الفرق أو الأفراد (Scott, 1991). ويتم إنشاء رسم بياني يسمى (sociogram) للعلاقات الاجتماعية بناءً على الإجابات التي تم الحصول عليها من استبيانات ركزت على الطريقة التي يتواصل الناس بها ومدى فاعلية هذا التواصل. يمكن أن تعرض هذه الرسوم البيانية العلاقات بشكل مرئي، ويمكن من خلالها استعراض أنماط عمليات التواصل والتغيرات التي حصلت عند نقاط حرجة، وهي تلك النقاط التي تطلب فيها وجود تواصل مكثف. يعرض الشكل 13-1 مثالاً على ذلك.



الشكل 13-1: أنماط التواصل بين فرق عمل في ثلاثة مواقع مختلفة

تمثل نقاط التقاطع في هذا الرسم البياني الأفراد، وتمثل الأطراف مسارات التواصل. تم رسم دوائر حول التجمعات للتظليل المرئي حول المناطق التي تعبر عن اتصالات مكثفة. ويعرض الرسم البياني للعلاقات الاجتماعية في الشكل 13-1 عدة نواحٍ مثيرة للانتباه، منها:

- الفريق الموجود في الولايات المتحدة مندمج بشكل أفضل من الفرق الموجودة في أوروبا وآسيا؛ ذلك يعني أن جميع أعضاء فريق الولايات المتحدة يتفاعلون مع بعضهم البعض.
- بعض أعضاء الفرق الموجودة في آسيا وأوروبا منعزلون ولا يتفاعلون مع معظم أعضاء فريقهم؛ وهذه الفرق هي 'N' و'L' في الفريق الموجودة في أوروبا، والفريق 'S' في الفريق الموجود في آسيا.
- الفريق الموجود في آسيا مجزأ؛ فعلياً، هناك مجموعتان جزئيتان، الأولى تتضمن 'O' و'P' و 'Q' و'R'، والثانية تتضمن 'Q' و'T' و'U' و'V'. ويمكن اعتبار العضو 'Q' حلقة وصل بين المجموعتين الجزئيتين.
- هناك تفاعل محدود بين الفرق الثلاث؛ ومحدود جداً بين الفريق الموجود في أوروبا والفريق الموجود في آسيا.
- يُعتبر العضو 'G' في الفريق الموجود في أوروبا حلقة الوصل بين جميع الفرق، لأن جميع عمليات التواصل بين الفريق الموجود في أوروبا والفرق الأخرى تمر من خلاله.
- إذا ترك العضو 'G' الفريق الموجود في أوروبا، يفقد هذا الفريق تواصله مع الفريقين الآخرين.

قد لا تكون هذه الملاحظات سيئة بالضرورة وذلك وفقاً للمهام المتعلقة بالمشروع وطريقة تخصيص المهام والفرق. على كل حال، عندما تكون عمليات التعاون والمشاركة في المعلومات حرجة ومهمة

لنجاح المشروع، توفّر عملية تحليل الشبكات الاجتماعية (SNA) أداة تحليلية قوية لاستخدامها من قبل المديرين. ويمكن أن تساعد عملية تحليل الشبكات الاجتماعية (SNA) مديري المشروع بالطرق الآتية :(Cross [et al.], 2002a)

- تعزز التعاون الفعال بين أعضاء الفريق الهام استراتيجياً: كما يبرهن الشكل 13–1، لدى الفريق الموجود في آسيا مجموعتين جزئيتين واضحتين. في كلا الفريقين اللذين يعملان في آسيا وأوروبا، كما لوحظ أنه يوجد أعضاء معينون موجودون على الحافة ولم يتواصلوا مع معظم أعضاء الفريق الذي ينتمون إليه. إذا كانت الفرق المطلوب منها أن تعمل كوحدة واحدة في نطاقها الخاص مهمة استراتيجياً، سوف يظهر تطبيق عملية تحليل الشبكات الاجتماعية هذه المشاكل في عمليات الدمج.
- إظهار نقاط التواصل الحرجة في التشابكات التي تظهر على شكل تقاطع وظيفي أو الحدود ذات المستويات المختلفة (الهرمية)، أو حدود جغرافية. يظهر الشكل 13–1 تفاعلات محدودة بين الفرق المختلفة:
  - تُظهر المجموعتان الجزئيتان الموجودتان في فريق آسيا تفاعلاً محدوداً، وقد يكون ذلك لأن كلاً منهما تمثل قسماً وظيفياً مختلفاً في هذا التنظيم. وأما في التنظيمات الكبيرة، فمن الطبيعي أن لا يعرف أي قسم أي معلومات عن عمل الأقسام الأخرى، وبالتالي عدم معرفة الطريقة الفعالة التي يمكن أن يعملوا بها معاً.
  - يوجد للتشتت المادي للفِرق من خلال وجودهم في حدود جغرافية مختلفة نفس التأثير. ويبدو هذا أكثر وضوحاً مع الفريق الموجود في آسيا، إذ إن تفاعل هذا الفريق مع الفريق الموجود في أوروبا ضعيف جداً، ولا يتفاعل على

الإطلاق مع الفريق الموجود في الولايات المتحدة. وقد تكون اختلافات اللغة والفروقات الثقافية وبُعد المسافة قد سببت ظهور هذا النمط السلوكي.

- على الرغم من أن التعاون بمستويات مختلفة لم يظهر في الشكل 13-1، يمكن إنشاء مخطط مشابه عن طريق إجراء مسح على مستوى مسؤولي الإدارة العليا؛ يمكن أن يُظهر هذا النوع من المخططات البيانية الطريقة التي يكون فيها مسؤولو الإدارة العليا مضطلعين في العلاقات غير الرسمية، وتكشف عملية تدفق المعلومات بالاتجاهين خلال هذا الفريق، وكيف أن القرارات التي يتخذونها تتأثر بهذه المعلومات. ويمكن تصور كيف أن القرارات قد تكون متحيزة لأن المعلومات التي يستقبلها أصحاب القرار تأتي من جهة محددة من التنظيم.
- التأكيد على وجود تكامل بين الفريق بعد إجراء خطوات باتجاه إعادة الهيكلية: يلجأ عادة كبار المسؤولين التنفيذيين إلى إجراءات إعادة هيكلة للتنظيم لجعله أكثر كفاءة وفاعلية. وقد أظهرت الأبحاث أنه ليس بالضرورة أن يتحسن الأداء نتيجة هذا الإجراء. بينما يعزى ذلك عادة إلى اختلال في البنية التنظيمية الرسمية أو إخفاق في عمليات القيادة، لكن العلاقات غير الرسمية تلعب نفس الدور هنا. في الحقيقة، يؤدي الاتجاه نحو البنى التنظيمية السطحية إلى وجود الفرق المتقاطعة وظيفياً وإنشاء تنظيمات أقرب من بعضها، ويؤدي إلى زخم أكبر في العمل ضمن العلاقات غير الرسمية أكثر منه ضمن البنية التنظيمية الرسمية المعلن عنها وعمليات العمل التفصيلية.

بغض النظر عن سبب كون مستوى التعاون أقل مما يجب، فسوف تُظهر عملية تحليل الشبكات الاجتماعية الأنماط الخاصة

بالتواصل، وذلك يمنح العمليات الإدارية الفرصة لبدأ تحليل المشكلة وطرح حلٍ مناسب. وتصبح هذه الأداة فعالة جداً كلما أصبحت التنظيمات أكبر وموزَّعة على نطاق واسع (تخيل وجود ثلاثة أو أربعة مستويات من التسلسل الهرمي في الهيكل التنظيمي، بعدد يزيد على ثلاثمئة شخص يتوزعون في مواقع مختلفة حول العالم). هذا تماماً ما يحدث في مشاريع التطوير البرمجي الموزَّعة. وفي هذه الظروف، من الصعب تقويم مستوى أداء العمل والقرارات التي تم اتخاذها دون الرجوع إلى أداة مشابهة لأداة تحليل الشبكات الاجتماعية.

فكرة مفيدة: قم بتسهيل عمليات المشاركة في المعرفة والتعاون بين الأعضاء عند تحولات معينة في المراحل.
ليس بالضرورة أن يكون جميع أعضاء تنظيم ما متصلين ببعضهم بعضاً بعلاقات شخصية غير (Cross [et al.], 2002b)؛ وذلك ليس بسبب التكلفة الباهظة وحسب، بل هذا سيؤدي أيضاً إلى ضغط كبير في تدفق المعلومات بين الأعضاء. ويمتلك الناس وقتاً محدوداً لاستثماره في تكوين علاقات جديدة والمحافظة عليها، لذلك يجب أن يركز المديرون على نقاط تحوُّل معينة عندما يكون هناك حاجة إلى المشاركة في المعرفة والتعاون. وفي حالة وجود المشاكل تم تشخيصها من خلال أداة تحليل الشبكات الاجتماعية، قد تكون بعض التقنيات مفيدة، مثل تقنية إنشاء مشاريع مشتركة من قبل موظفين يعملون في فرق مختلفة، وتقنية إنشاء منتديات جديدة من أجل عمليات التواصل كالاجتماعات الأسبوعية أو استخدام لوحات الرسائل الإلكترونية، ومنح مديري الفرق المشاركة حوافز مادية.

فكرة مفيدة:
انتبه إلى التغيرات المفاجئة في زخم التواصل.
قد تكون التغيرات المفاجئة في زخم التواصل مؤشراً تحذيرياً مبكراً للنقاط الساخنة في المشروع، إذ قد تكون مؤشراً على أن المشاكل قد بدأت، وقد يدل هذا على نقص في عمليات التواصل، أو وجود اتصالات سيئة بين الفرق المشاركة. وفي كلا الحالتين، لم يتم إيصال المعلومات المطلوبة كاملة أو لم يتم فهم هذه المعلومات بشكل جيد.

13-5 الملخص والاستنتاجات

ناقش في هذا الفصل أدوات التحكم بالتواصل والعوائق التي تواجهها والأنشطة المهمة والحرجة لتحسين أداء تطوير المنتج. إن تنسيق عملية التواصل هي عملية مهمة أيضاً لأنه يكون لدى الفرق مستوى أفضل من الأداء. كما نوقشت المبادئ المستمدة من نظرية التنظيم، مثل العلاقات الترابطية التنظيمية وآلياتها، وأساليب التواصل والعقول التجميعية التي تساعد في تعزيز التعاون الفعال بين الفرق المتعاونة. قد يؤدي التواصل غير المنضبط إلى تدفق فائض للمعلومات ووجود معلومات غير صحيحة. وقد تم تقديم تحليل للشبكات الاجتماعية كوسيلة لتتبع التغير في زخم التواصل للوصول إلى إدارة فعالة للاتصالات للفرق المتعاونة.

13-6 أسئلة للمناقشة

1. أذكر بعض الأدوات المهمة الخاصة بالتحكم بالتواصل في مشاريع تطوير البرمجيات الموزَّعة المراكز.

2. أذكر وناقش بعض العوائق التي تواجه التواصل.
3. كيف يمكن الوصول إلى تنسيق أفضل للتواصل بين الفرق المتعاونة؟
4. أذكر بعض التقنيات التي تساعد على مراقبة والتحكم الفعال بالتواصل.

المراجع

Books

Allen, Thomas John [et al.]. *Managing the Flow of Technology: Technology Transfer and the Dissemination of Technological Information within the R&D organization*. Cambridge, MA: MIT Press, 1984.

Scott, John. *Social Network Analysis: A Handbook*. Second Edition. Thousand Oaks, CA: Sage Publications, 1991.

Periodicals

Churchill, Elizabeth F. and Christine Halverson. «Social Networks and Social Networking.» *IEEE Internet Computing*: vol. 9, no.5, 2005, pp. 14-19.

Cross, Rob, Nitin Nohria and Andrew Parker. «Six Myths about Informal Networks and How to Overcome them.» *Sloan Management Review*: vol. 43, no. 3, 2002b, pp. 67-75.

———————, Stephen P. Borgatti and Andrew Parker. «Making Invisible Work Visible: Using Social Network Analysis to Support Strategic Collaboration.» *California Management Review*: vol. 44, no. 2, 2002a, pp. 25-46.

Gallagher, Séamus and Ian R. McChesney. «Communication and Co-ordination Practices in Software Engineering Projects.» *Information and Software Technology*: vol. 46, no. 7, 2004, pp. 473-489.

Kraut, Robert E. and Lynn A. Streeter. «Coordination in Software Development.» *Communications of the ACM*: vol. 38, no. 3, 1995, pp. 69-81.

Malone, Thomas W. and Kevin Crowston. «The Interdisciplinary Study of Coordination.» *ACM Computing Surveys*: vol. 26, no. 1, March 1994, pp. 87-119.

Sosa, Manuel E. [et al.]. «Factors that Influence Technical Communication in Distributed Product Development: An Empirical Study in the Telecommunications Industry.» *IEEE Transactions on Engineering Management*: vol. 49, no. 1, February 2002, pp. 45-58.



القسم الخامس

دراسات الحالة



الفصـــــل الرابع عشر

مشروع الأستديو العالمي 2005

حتى هذا اليوم، نفّذت شركة سيمنز عدة مشاريع تطوير برمجيات باستخدام فرق موزَّعة، وهي بالتالي مدركة أن الأسلوب الموزع لتطوير البرمجيات يؤثِّر بشكل كبير على جميع مراحل العمل المتعلق بالبرمجة، ويشمل هذا التأثير الممارسات الإدارية (Herbsleb [et al.], 2005). وهناك تحديات كبيرة تتعلق بالتطوير البرمجي بالأسلوب الموزع عالمياً. ويركز مركز البحوث في شركة سيمنز جهوده لتكوين فهم أفضل لهذه التحديات وتطوير ممارسات تساعد في الإدارة الناجحة لمشاريع تطوير البرمجيات الموزَّعة المراكز. فقد باشر المركز في مشروع بحثي تطبيقي يدعى مشروع الأستديو العالمي يسعى لتحقيق الأهداف الآتية:

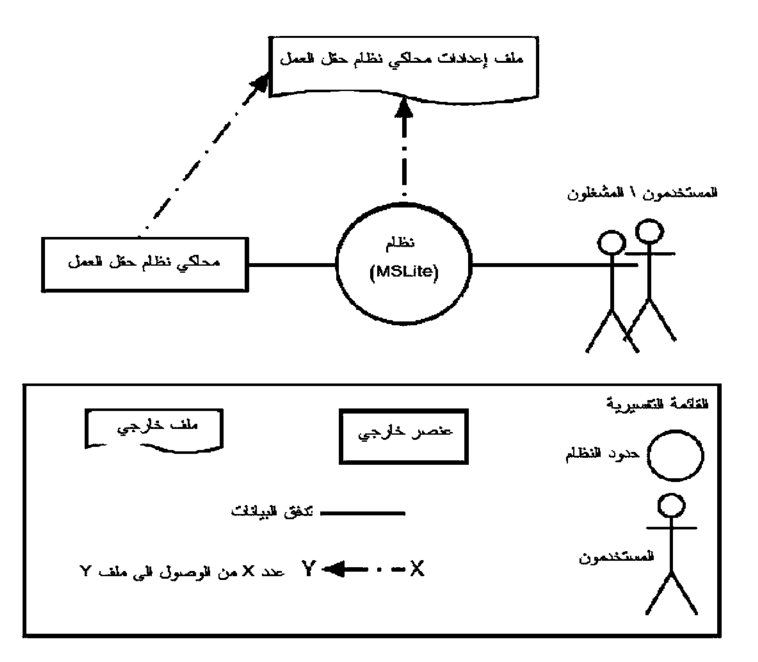
1. تطوير برمجية (MSLite) بشكل ناجح بالتعاون مع فرق من طلاب ست جامعات من جميع أنحاء العام. ولتحقيق ذلك، يتم توثيق العمليات وأفضل الممارسات المستخدمة، وكذلك القضايا التي تتم مواجهتها.

2. جمع وتحليل البيانات من هذا المشروع من أجل فهم وتدوين مدى فعالية عملية تطوير البرمجيات الموزَّعة المراكز. تم تحديد تفاصيل البيانات التجريبية التي سيجمعها فريق من الباحثين يعملون في مركز البحوث في شركة سيمنز، الجامعات المشاركة في تطوير برمجية (MSLite) هي: جامعة ولاية بنسلفانيا ومعهد هندسة البرمجيات في جامعة كارنيغي ميلون وجامعة هارفارد للأعمال.

كانت فرق الطلاب المشاركين من الجامعات: جامعة كارنيغي ميلون، والمعهد الدولي لتكنولوجيا المعلومات في بانغالور (IIITB)، وجامعة موناوث (MU)، وجامعة ميونخ التقنية (TUM)، والجامعة البابوية الكاثوليكية في ريو جراند دو سول (PUCRS)، وجامعة ليمريك (UL).

14-1 مشروع (MSLite)

يهدف مشروع (MSLite) إلى إنشاء مركز إداري موحد لبناء الأنظمة الأوتوماتيكية ـ كأنظمة التدفئة وأنظمة التكييف (HVAC) والتحكم بالوصولية والإضاءة ـ والتي تمكّن المسؤول عن المنشأة من تشغيل مثل هذه الأنظمة. وكمقياس عملي، تم استبدال أنظمة الأبنية التلقائية بمكوّنات داخلية – محاكي نظام حقل العمل (FSS)– والذي يقوم بمحاكاة المجالات التلقائية للمبنى. يظهر الشكل 14-1 وصفاً لمخطط نظام (MSLite).



الشكل 14–1: سياق المخطط لنظام MSLite

في ما يأتي ملخص للمتطلبات الوظيفية المهمة لنظام (MSLite):

1. التعامل مع المكوّنات في الميدان (المكوّنات المتعلقة بمجال العمل التلقائي في البناية، على سبيل المثال، مجسات أنظمة التدفئة وأنظمة التكييف) التي تم تمثيلها في محاكي نظام حقل العمل.
2. إصدار الأوامر لعناصر الحقل من خلال محاكي نظام حقل العمل لإجراء تغيير قيم تتعلق بخصائص هذه العناصر عن طريق استقبال أمر «تغير القيمة» (change-of-value COV) من محاكي نظام حقل العمل ليقوم بتحديث غير متزامن لحالة هذه المكوّنات كما تم تمثيله في واجهة التطبيق.

3. تحديد الشروط المنطقية اعتماداً على قيم خصائص عناصر الحقل؛ وحين يتم تحقق هذه الشروط، يتم حدوث تفاعلات وبالتالي إصدار أوامر لعناصر الحقل.

4. تحديد الشروط التحذيرية بشكل مشابه للشروط المنطقية؛ وحين يتم تحقق هذه الشروط، يتم انطلاق تحذير لينبه المستخدم المناسب. وتستمر فعالية هذه التحذيرات لفترة زمنية محددة إلى أن يتعامل معها المستخدم ذو الصلاحيات المناسبة.

تم إنشاء بنية مشروع (MSLite) بحيث يحاكي نموذج (المحور والنطق)، ويدعى عادة نموذج «العمل الموسع» في شركة سيمنز. و«المركز» هو الفريق المركزي في مركز البحوث في شركة سيمنز الموجود في برينستون، والذي يقوم بتنظيم عمل بعض أعضائه من الطلاب المتدربين. و«الفروع» هي الفرق التي تعمل عن بُعد، وتتكون الفرق من مجموعة من طلاب جامعات مختلفة موزَّعة في مناطق مختلفة في جميع أنحاء العالم. ويكون الفريق المركزي مسؤولاً عن المتطلبات وهيكلية البرمجيات وبعض الجوانب المتعلقة بالتصميم واختبار النظام والدمج وإدارة المشروع وتحديد العمليات بشكل كلي. وأما الفريق الذي يعمل عن بُعد فيكون مسؤولاً عن التصميم والبرمجة والاختبارات التلقائية للوحدات للوصول إلى مهام معرّفة بشكل جيد وفقاً للوحدات البرمجية أو الأنظمة الفرعية التي يحددها الفريق المركزي خلال مراحل تحديد الهيكلية وعمليات التخطيط للمشروع. ويتم إدارة العلاقات بين الفريق المركزي والفرق التي تعمل عن بُعد عن طريق وظيفة مدير التزويد. ويوجد كذلك مدير تزويد لكل فريق يعمل عن بُعد، ويكون من يشغل هذا المنصب عضواً في الفريق المركزي.

هناك ثمة معلومات إضافية عن مشروع الأستديو العالمي تشمل

لقطات من نظام الـ Wiki الذي استخدمته فرق الطلاب، تم إضافتها في القرص الضوئي المرفق.

14-2 التحديات التي تم مواجهتها في السنة الأولى من تطوير نظام (MSLite)

امتدت السنة الأولى لهذا المشروع من تموز/ يوليو 2004 إلى أيار/ مايو 2005. وقد تم تنظيم نظام (MSLite) على شكل إصدارات هندسية زمنية متتابعة تتألف من مجموعة من الوظائف التي من المفترض أن تكون مستقلة عن بعضها بعضاً. وكانت خطة التطوير تزايدية، وذلك عن طريق إضافة وظائف جديدة في كل إصدار زمني هندسي. وتم إبقاء الإصدارات الهندسية تحت نظام التحكم بالإعدادات، وأُتيحت لمجموعة الاختبارات الداخلية في الفريق المركزي لمركز البحوث في شركة سيمنز لإجراء اختبار النظام عليها. وتشتمل المهام التي يسلِّمها في كل إصدار زمني كل فريق يعمل عن بُعد على بيانات العمل ومواصفات تفصيلية للمتطلبات والمهام الهيكلية والتصميمية واختبارات الوحدات (الخطط، الحالات، والتقارير) وصيغة برمجية موضح عليها تعليقات مناسبة. ويجب أن يتم تحديد جميع هذه المهام مسبقاً حين تسليم مكوّنات ما تم تخصيصه للفريق الذي يعمل عن بُعد.

فيما يُظهر تطوير البرمجيات الموزَّعة المراكز مستقبلاً واعداً، لكنه أيضاً يواجه تحديات؛ وإذا لم يتم التعامل مع بعض هذه التحديات في الوقت المناسب فقد لا يمكن التغلب عليها. وقد كانت الأمور التي واجهت مشروع (MSLite) نموذجية لما صنفه هيرسليب ومويترا (Herbsleb and Moitra, 2001) بالأبعاد المتعددة لتطوير البرمجيات الموزَّعة المراكز:

- **أمور استراتيجية** : لا يوجد حل مثالي لعملية تقسيم البرنامج إلى مهام عمل تأخذ بعين الاعتبار توفّر المصادر ومستوى الخبرات التقنية واقتران الوحدات البرمجية. وكان الهدف من المخطط المخصص للعمل في مشروع (MSLite) بسيطاً جداً، وفي نواحٍ أخرى لم يضع بعين الاعتبار المهارات التقنية لدى الفرق التي تعمل عن بُعد والعلاقات الترابطية بين عناصر الهيكلية.
- **أمور ثقافية**: تؤثِّر الثقافة في أمور مهمة وحرجة كالحاجة إلى بنية تحتية وأسلوب إنشاء الهيكل التنظيمي والتعامل مع الوقت وطرق التواصل (Hofstede, 1997). وقد أصبحت هذه الفروقات عقبة تواجه التواصل الفَّعال، كما كان واضحاً في مشروع الأستديو العالمي الذي احتوى على ست فرق تعمل عن بُعد تتوزع في خمس دول في أربع قارات.
- **قضايا التواصل**: تحتاج عملية تطوير البرمجيات، وبشكل خاص في المراحل الأولية، إلى الكثير من عمليات التواصل (Perry [et al.], 1994). ويؤدي غياب المحادثات إلى وجود مفاجآت من المواقع المتباعدة، وهذا يؤدي إلى وجود اختلال في العمل والحاجة إلى إعادة صياغته. فبدلاً من الدردشات السريعة وجهاً لوجه، لجأت فرق مشروع (MSLite) إلى إضافة آلية مخصصة للتواصل؛ فكان هناك تكرارات للمواضيع التي يتم فيها التواصل مع الفريق المركزي ما أدَّى إلى وجود ضغط عليه؛ وقد تم استخدام وسائل وأساليب مختلفة للتواصل بين الفريق نفسه وفي أوقات مختلفة، وبالتالي فقد التنسيق لعملية التراسل. وقد أثَّر هذا سلباً في قدرة التنسيق والتحكم. وأصبح من الصعب جعل أنشطة الفرق المختلفة متزامنة بسبب عدم القدرة على الحصول على جهد مُوجَّه باتجاه واحد ومعلومات دقيقة عن الوضع الحالي للعمل.

- **إدارة المعرفة**: عدم وجود آلية فعَّالة للمشاركة بالمعلومات وسوء الحفاظ على عمليات التوثيق وعدم وجود تعاون للمهام، ويؤدِّي كل هذا إلى تفاقم مشكلة إدارة المعرفة بطريقة تُؤثِّر سلباً في الإمكانيات الفعلية لمشاريع تطوير البرمجيات الموزَّعة المراكز. وكانت الصعوبات التي واجهت الفرق التي تعمل عن بُعد في فهم المتطلبات والنية التصميمية دليلاً على ذلك، فهم لم يشاركوا في إعداد المتطلبات والهيكلية. فقد كانت المواصفات الموضوعة للمتطلبات والهيكلية، من وجهة نظرهم، مختصرة جداً.
- **أمور تتعلق بإدارة المشروع والعمليات**: عدم وجود تزامن، وبشكل خاص عدم وجود معالم ومعايير واضحة لنقاط البداية والنهاية في المهام الموكلة، ما يفاقم المشاكل في مشاريع تطوير البرمجيات الموزَّعة المراكز. وتتمتع الفرق التي تعمل عن بُعد بالحرية الكاملة في تحديد العمليات ومعايير معالم العمل، ما يؤدي إلى ظهور صعوبات في مسار العمل.
- **أمور تقنية**: وجود تناقضات في قدرات وإعدادات الشبكات وصيغ غير متوافقة للبيانات وإصدارات مختلفة للأدوات نفسها هي بعض ما يواجه الأمور التقنية عادة. وقد تُرك في مشروع الـ (MSLite) لكل فريق يعمل عن بُعد الحرية في إعداد بنية تحتية خاصة به. ولم تكن أدوات البرمجة المتوفّرة لهذه الفرق تدعم عمليات البرمجة في مواقع مختلفة.

14-3 المنهجية للسنة الثانية من تطوير نظام (MSLite)

يواجه التعامل مع الأمور المتعلقة بالتطور الموزع مجموعة من التحديات، وتم ملاحظة معظم هذه التحديات في مشروع (MSLite) خلال السنة الأولى. وبشكل أساسي، عندما تم تشغيل طلاب مازالوا

على مقاعد الدراسة، احتاج الفريق المركزي إلى بعض الوقت من أجل تجربة هذه الأمور وتطوير البنية التحتية الضرورية وإنشاء مهام البرمجة الموكلة إلى الفرق التي تعمل عن بُعد. وتقوم الأقسام الآتية من هذا الفصل بتفصيل الاستراتيجية المتَّبَعة في السنة الثانية.

14-3-1 سير العمل

قام الفريق المركزي بتحديد الوظائف والمهام لكل من الفريق المركزي والفريق الذي يعمل عن بُعد. ويتكون الفريق المركزي من الوظائف الآتية (مع بعض خصائص مسؤوليات هذه الوظائف):

- **مهندس المتطلبات**: يحتفظ بالمتطلبات الوظيفية المهمة ومحددات طرق الاستخدام ومواصفات واجهة التطبيق؛ كما يطابق ويتأكد من صحة مواصفات التصميم التي أنجزها الفريق الذي يعمل عن بُعد، ويحافظ على القدرة على التتبع في هذه المهام.
- **مهندس التصميم**: يعمل على صيانة هيكلية وتصميم النظام؛ ويتأكد من عملية نقل المواصفات للفريق الذي يعمل عن بُعد (يتضمن ذلك التأكد من الفهم الصحيح لهيكلية النظام التصميمية)؛ يحدد الحلول والمشاريع ومساحات التسلسل الهرمي للتقنية المستخدمة في عمليات التطبيق.
- **مهندس عمليات الدمج والتكامل (مسؤول عمليات البناء البرمجي)**: يحافظ على مستودع تخزين الشيفرة البرمجية وبيئة إنشاء الوحدات المبرمجة تلقائياً؛ ينفذ الاختبارات المتعلقة بالدمج ويعمل على صيانتها، كما يتأكد من أن الفريق الذي يعمل عن بُعد قد نفَّذ عمليات الدمج.
- **مدير البنية التحتية**: يحافظ على البنية التحتية ويدير جميع الأمور المتعلقة بالبنية التحتية التي تطرأ في جميع الفرق.

- **مسؤول نظام Wiki**: يحافظ على تصميم وبنية نظام Wiki، ويحافظ على اللباقة في استخدام نظام Wiki. ويشمل ذلك اختبار وتدقيق جميع نواحي نظام Wiki (يوجد المزيد عن وظيفة نظام Wiki في القسم الآتي).
- **مدير التزويد**: يدير جميع التعاملات مع الفريق الذي يعمل عن بُعد في ما يتعلق بالمهام الموكلة إليهم؛ التأكد من أن جميع عمليات التواصل تتم باتباع أسلوب مناسب، ويقوم بتدريب إداريي المشروع وتقنيي التحكم والسيطرة في الفرق التي تعمل عن بُعد. وتُمثِّل هذه الوظيفة حلقة الوصل الأساسية بين الفريق المركزي والفريق الذي يعمل عن بُعد.
- **مدير ضمان الجودة**: ينشئ اختبارات القبول لكل مرحلة عمل ويتأكد من أن مراحل العمل تخضع للمعايير الموضوعة؛ يتحقق من صحة اختبار وحدة الحالات والتقارير التي أصدرتها الفرق؛ يتأكد من أن العناصر الثابتة في الهيكلية قد تمَّتْ مراجعتها بشكل مناسب؛ ويتأكد من التجاوب مع التقارير حول العيوب وعملية تتبعها.
- **مدير سير العمل**: ينشئ ويحدّث عملية تحديد الوظائف والمسؤوليات كلما تطلب الأمر؛ يحدد العمليات الجديدة ويحافظ على العمليات الحالية في قسم العمليات في نظام Wiki؛ ويتأكد من أن الفريق المركزي والفرق التي تعمل عن بُعد تلتزم بالعمليات التي تم تحديدها.
- **مدير المشروع**: يحافظ على خطة المشروع ويقوم بتطبيقها.

في ما يأتي وظائف الفرق التي تعمل عن بُعد وبعض صفات المسؤوليات لهذه الوظائف:

- **المبرمج**: يكمل جميع المهام (المتطلبات والتصميم وعمليات البرمجة) كما تم تحديدها في البرنامج الزمني للمشروع ويحقق

متطلبات الجودة؛ التأكد من أنه قد تم الالتزام بالتصميم المقرر أثناء عملية البرمجة؛ كتابة وتنفيذ الاختبار التلقائي للوحدات البرمجية؛ الالتزام بتنفيذ عمليات الدمج والتكامل. إن جميع أعضاء الفريق الذي يعمل عن بُعد هم مبرمجون في جميع الأوقات.

- **قائد الفريق**: يتعامل مع جميع الأمور التي تتعلق بالعلاقة مع مدير التزويد؛ يتأكد من أن الفريق يكمل جميع المهام كما هو مخطط لها في خطة المشروع، والالتزام بالوقت، والتأكد من تحقق متطلبات الجودة.
- **مهندس المتطلبات**: يسهّل فهم الفريق لجميع المتطلبات؛ يتأكد من أن جميع مواصفات المتطلبات الموكلة إلى الفريق قد تم تسليمها في المواعيد المحددة.
- **مدير المصممين**: يُسهّل فهم الفريق لمواصفات الهيكلية والتصميم؛ والتأكد من أنه تم تسليم جميع مواصفات العمل الموكل إلى الفريق قد تم تسليمها في المواعيد المحددة.
- **مدير ضمان الجودة**: مسؤول عن ضمان جودة كل ما يتم تسليمه من البرمجيات وجميع المهام الأخرى أيضاً؛ كما إنه مسؤول عن تحقيق معايير الجودة التي تم تحديدها.
- **مدير البنية التحتية**: مسؤول عن إعداد البنية التحتية للفريق والمحافظة عليها.
- **مدير إدارة العمل والتوثيق**: يطبق محددات جميع وظائف الفريق الذي يعمل عن بُعد؛ ويتأكد من استخدام نظام Wiki بالطريقة المناسبة ـ سوف يكون مسؤولاً عن جميع المهام الخاصة بنظام Wiki التي يسلمها الفريق.

يقوم الفريق الذي يعمل عن بُعد بتحديد شاغلي هذه الوظائف في بداية عملهم. كما يستطيعون تغييرهم في أي وقت خلال

المشروع (لأنهم ما زالوا في المجال الأكاديمي كطلاب، ويتطلب ذلك التعرض إلى وظائف مختلفة)، ولكن لا بد من إبلاغ الفريق المركزي بهذه التغييرات التي تُجرى في بنية الفريق.

حدد الفريق المركزي أيضاً العمليات التي تعالج جميع الأمور في المشروع وفي التنظيم ككل:

- **عملية إدارة التهيئة**: تصف طريقة استخدام مستودع الوثائق والشيفرة البرمجية المصدرية في موقع الفريق المركزي.
- **عملية التواصل**: تصف أهداف واستخدامات أنواع التواصل المختلفة المتوفّرة، كنظام Wiki والقوائم البريدية واتصالات المؤتمرات.
- **عملية التصميم والبرمجة**: تصف الخطوات المتبعة لعمليات التصميم والبرمجة منذ بدايتها عندما يصدر الفريق المركزي النسخة الأولية لخطة البرمجة إلى نهايتها عندما يقوم الفريق الذي يعمل عن بُعد بتسليم المهام المتوقعة منه، والتي وافق عليها مدير التزويد. ويشمل ذلك المفاوضات التي تتم حول خطة البرمجة الموضوعة لمكوّنات نهاية كل مرحلة عمل، والمهام التي يجب أن ينجزها كل فريق يعمل عن بُعد خلال فترة زمنية معينة، والمعايير التي يضعها مدير التزويد لقبول ما يتم تسليمه من الفريق الذي يعمل عن بُعد.
- **عملية الدمج**: تشمل هيكلية المستودع وبيئة عملية الدمج التلقائي والطريقة التي تعمل بها وإمكانية استخدامها من قبل الفرق التي تعمل عن بُعد. وتقوم بتصنيف التعديلات إلى تعديلات مستقلة وتعديلات مترابطة، والكيفية التي يجب أن يتعامل بها المبرمجون في الفريق الذي يعمل عن بُعد ومسؤول عمليات الدمج في الفريق المركزي مع هذه التعديلات.

- **عملية إدارة التغيير**: تحدد العملية التي يتم من خلالها طلبات إجراء تعديل وإظهار تأثيراتها وضمان مستوى معينً من القدرة على تتبعها، ويشمل ذلك خطوات عمل التعامل مع التعديلات باستخدام أداة أوتوماتيكية تشتمل على آلية لأخذ آراء أصحاب القرار حول الموافقة على إجراء التغييرات، بافتراض أن «التغييرات هي تغييرات إيجابية». وتؤيد تغييرات على المتطلبات، والهيكلية، ومواصفات التصميم التي يزود بها الفريق المركزي، أي نواحٍ قد تؤثر في عملية التصميم التي تقوم بها الفرق.
- **عملية الإبلاغ عن العيوب وتتبعها**: تصنيف العيوب إلى عيوب تشغيلية وعيوب برمجية وعيوب في الوثائق، كما تتضمن وصف آلية إرسال العيوب، وتوكيل العيوب إلى المبرمجين، وإصلاح العيوب، والتحقق من صحتها باستخدام أداة تقوم بذلك تلقائياً.
- **عملية الاجتماعات**: تضع الخطوط العريضة لتنظيم الاجتماعات المتلفزة الأسبوعية بين الفرق التي تعمل عن بُعد ومدير التزويد الذي يعمل في الفريق المركزي. وتتضمن عملية التنظيم هذه إعداد جداول عمل الاجتماعات التي يقوم الفريق الذي يعمل عن بُعد بكتابتها. وتصف هذه العملية أيضاً ما يجب عمله قبل الاجتماع وخلاله وبعده أيضاً لتأكيد فعالية هذه الاجتماعات.
- **عملية إدارة المخاطر**: تحدد القواعد المستخدمة في تتبع المخاطر التي تُؤثِّر على المشروع والتعامل معها. ويُتوقع من جميع أعضاء الفريق المركزي وأعضاء الفريق الذي يعمل عن بُعد إضافة المخاطر التي يتوقعونها إلى النماذج. ويُتوقع من الفريق المركزي استخدام هذه النماذج كدليل لاتخاذ الإجراءات المناسبة لتعديل ذلك.

- **عملية تقويم أداء الفريق**: يبين كيفية تقويم أداء الفريق أسبوعياً؛ وعلى الرغم من أن تقويم الأداء هذا سيبقى داخلياً في الفريق المركزي، فسيتم إعلام ومكافأة الفريقين ذوَي التقويم الأعلى.

14-3-2 التعاون والتواصل وإدارة المعرفة

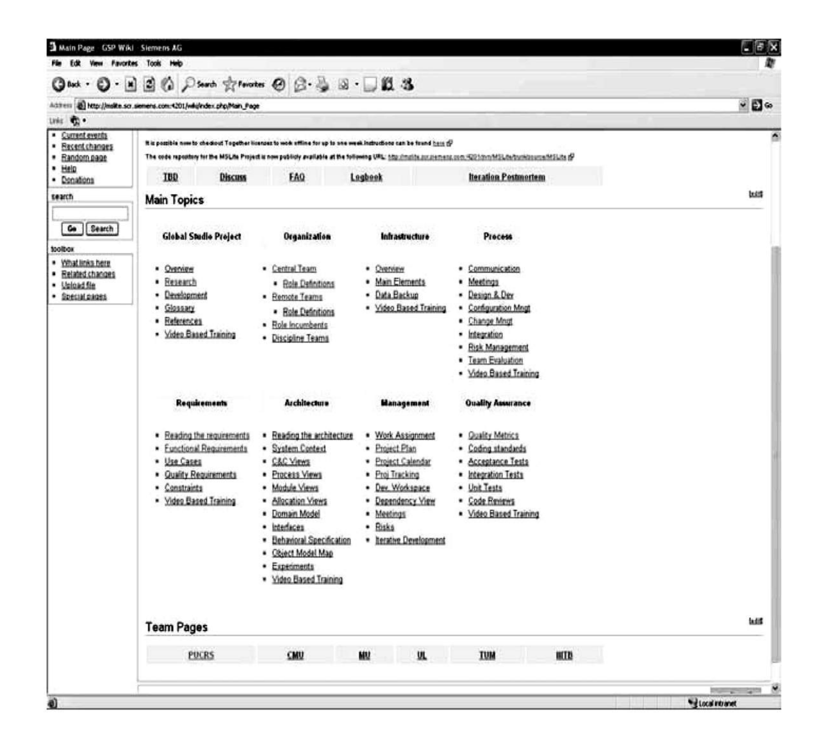
كانت الجهود المبذولة في إرساء التعاون والتواصل هي الجانب الأضعف في إنجازات العام الأول. ولم يكن هناك بنية تحتية متوفّرة بشكل خاص للمعلومات والمعرفة. ويستخدم الفريق المركزي الآن نظام Wiki في المشروع، وهو عبارة عن تطبيق من تطبيقات شبكة الإنترنت يتيح للمستخدمين إضافة محتويات كما في منتديات الإنترنت، ويتيح كذلك لأي كان إجراء تعديلات على هذه المحتويات. وقد قرر الفريق المركزي استخدام تطبيق (Media Wiki) (الشكل 14-2) (http://meta.wikimedia.org/wiki/MediaWiki).

في ما يأتي بعض الطرق التي يمكن استخدام نظام Wiki فيها لتحقيق التعاون والتواصل وإدارة المعرفة:

- **المستودع المركزي للوثائق**: نظام Wiki هو المستودع الوحيد لجميع الوثائق المرتبطة بالمشروع والعمل الموكل إلى الفرق التي تعمل عن بُعد. تتم إضافة القسم المتعلق بالفريق المركزي (تحت مسمى «المواضيع الرئيسة») والحفاظ عليه بصرامة بواسطة مسؤول نظام Wiki. ونعرض في هذا القسم النظام الذي يتم تطويره بشكل كامل ودقيق ومحدث أولاً بأول.
- **مستودع الوثائق الخاصة بمواقع العمل البعيدة**: نظام Wiki هو أيضاً المستودع الوحيد لجميع وثائق الفرق التي تعمل عن بُعد. ويتم تشجيع الفرق التي تعمل عن بُعد على إنجاز مهامهم

المتوقعة منهم والتعاون من خلال المساحة المتوفّرة لهم في «قسم صفحات الفريق». وتستطيع الفرق التي تعمل عن بُعد كذلك استخدام هذه الصفحات كمكان مخصص لإنشاء الوثائق المطلوبة من قبل الفريق المركزي قبل أن يتم نقلها إلى مكانها الصحيح في صفحات الفريق المركزي. ويضمن هذا أيضاً وضوحاً كاملاً لإنجازات جميع الأعضاء في أي مرحلة من المراحل.

- **المناقشات**: تحتوي كل صفحة من صفحات نظام Wiki على صفحات محادثة مرتبطة بها. ويستخدم الفريق المركزي هذه الصفحات لجمع الأسئلة والحصول على الآراء والانطباعات، وأي أمور أخرى يتطلبها من الفريق الذي يعمل عن بُعد. وقد يتم إرسال هذه الأمور مباشرة عن طريق مدير التزويد أو في الاجتماعات الأسبوعية مع الفريق الذي يعمل عن بُعد. ويقوم الفريق الذي يعمل عن بُعد بتوثيق الإجابات أو التوضيحات عن تساؤلاتهم. ويمنح هذا الفريق الذي يعمل عن بُعد إمكانية طرح تساؤلاته بشكل فوري وربطها مع مهام خاصة في نظام Wiki. كما يقومون أيضاً بإجراء نقاشات مع فرق أخرى تعمل عن بُعد لتوضيح القضايا أو التوجيه، وبذلك يتم الحد من قنوات التواصل التي تنشأ عن الإجابة عن نفس التساؤلات لكل فريق يعمل عن بُعد.
- **ما ينبغي عمله** (To Be Done): عندما تكون إحدى المهام أو قسم منها غير مكتمل، يمكن الإشارة إلى ذلك بسهولة عن طريق علامة (ما ينبغي عمله TBD) معرفة مسبقاً. وتظهر هذه المهام بشكل تلقائي في قسم «ما ينبغي عمله TBD»، ويمكن الدخول إليه من الصفحة الرئيسة في نظام Wiki.



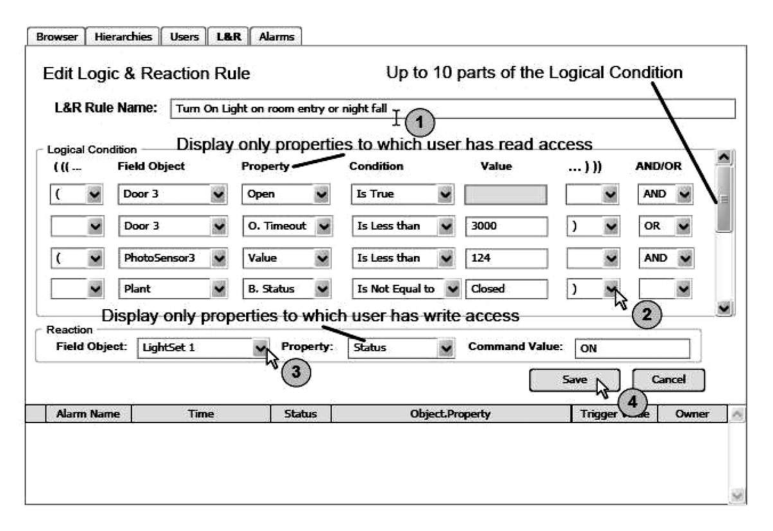
الشكل 14–2: الصفحة الرئيسة في نظام (MSLite Wiki)

- **إعلانات نظام** Wiki: حين يتم إنجاز تعديلات مهمة أو جذرية، يعلن الفريق المركزي عن هذه التعديلات في قسم «جديد وجدير بالملاحظة» «new & noteworthy» في الصفحة الرئيسة في نظام Wiki. ويتم إعلان هذه التعديلات لجميع الفرق ذات العلاقة عن طريق البريد الإلكتروني. وتُستخدم هذه الخاصية في حال إجراء تعديلات على نظام Wiki، وليس التعديلات التي تؤثر في المتطلبات أو الهيكلية أو مواصفات التصميم، التي يتم التعامل معها عن طريق عملية إدارة التعديلات.

- **سجل الأحداث**: يسهل سجل الأحداث عملية طرح المشاكل والأمور الأخرى المتعلقة بعملية البرمجة بشكل صادق ومنفتح. ويتيح نموذج بسيط وسهل الاستخدام المجال أمام جميع المشاركين بالمشروع لسرد المشاكل التي يواجهونها أو المجالات التي يشعرون أنها بحاجة إلى التطوير.
- **تحليل عملية التشغيل بعد إتمامها**: في نهاية كل مرحلة، يعقد مديرو التزويد تحليلاً لعملية التشغيل بعد إتمامها (post-mortem) مع فرقهم لمناقشة الطريقة التي تم بها التخطيط للعمل وتنفيذه، وكيف يمكن الاستفادة من المرحلة الأولى في المراحل اللاحقة. ويتم عرض هذه المناقشات في قسم عنوانه «ما بعد الفترة البرمجية التكرارية» في الصفحة الرئيسة من نظام Wiki.

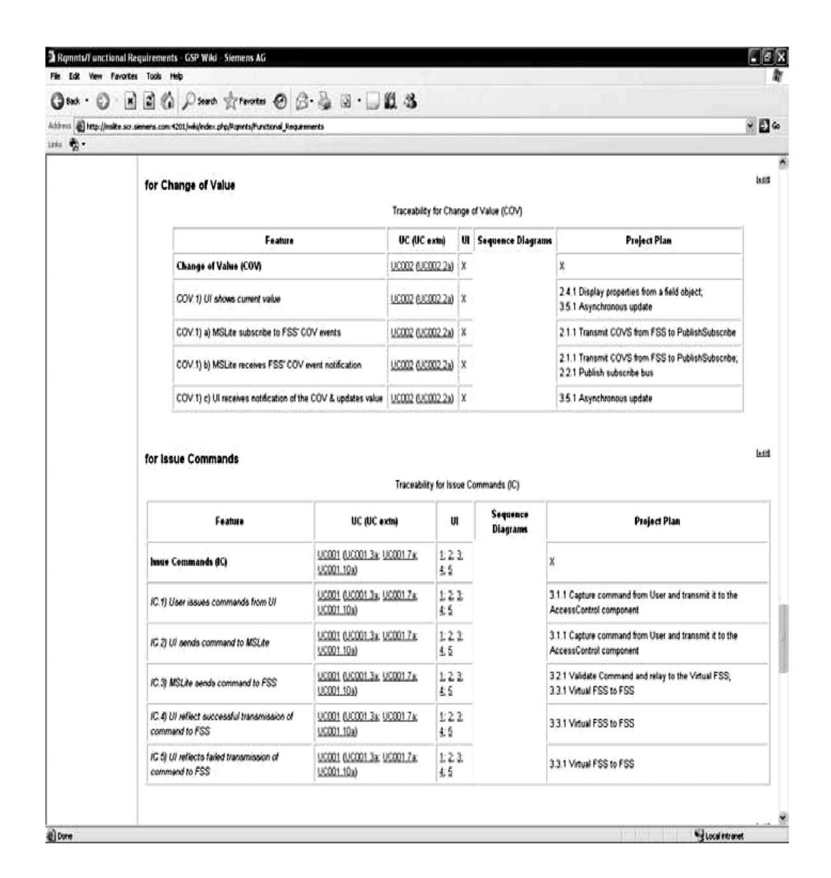
14-3-3 المتطلبات

إن نظام Wiki هو الوسيط الأساسي لتحديد مواصفات المتطلبات بما في ذلك المتطلبات الوظيفية المهمة ومحددات حالات الاستخدام ومواصفات واجهة التطبيق. وتكون المتطلبات الوظيفية المهمة على شكل جمل باللغة الانجليزية تقترح الوظائف الأساسية للنظام المتاح للمستخدم أو المستخدمين. ويتم تجميعها في مجموعات منطقية مختلفة وفقاً لترابطها مع هذه المتطلبات. وتتضمن مواصفات حالات الاستخدام مخططات لحالات الاستخدام في تطبيق (Borland Together) ومقسمة إلى رزم منطقية (وفقاً لعمليات التجميع في المتطلبات الوظيفية المهمة)، كما تتضمن وصفاً لحالات الاستخدام باستخدام النموذج التقليدي للعملية المنطقية الموحدة.



الشكل 14-3: مثال على مواصفات واجهة المستخدم

يتم ربط خطوات حالات الاستخدام المرتبطة بالوظائف الظاهرة في واجهة التطبيق بشكل تشعبي مع مواصفات واجهة التطبيق، وهي عبارة عن نماذج الشاشة (يتم إنشاؤها باستخدام تطبيق Microsoft (MS) Visio) مع وضع علامات على الخطوات التي تشير إلى ترتيب الخطوات التي قام بها المستخدم على واجهة التطبيق لإنجاز إحدى الوظائف. يتيح نظام Wiki محافظة الفريق المركزي على الترابط بين هذه المهام المتعددة بطريقة بسيطة وقابلة للاستخدام بأقل جهد ممكن على الفريق المركزي. **أولاً**، يتم تصنيف المتطلبات الوظيفية المهمة إلى مجموعات مختلفة؛ يتم تفصيل كل متطلب إلى مواصفات وظيفية؛ يتم ربط كل مواصفة وظيفية مع مواصفات حالة استخدام واحدة أو أكثر؛ يتم ربط خطوات حالات الاستخدام بدورها إلى أجزاء مواصفات واجهة التطبيق بالطريقة المناسبة. ويتم أيضاً الحفاظ على القدرة على تتبع المتطلبات الوظيفية المهمة عن طريق مخططات متسلسلة وعن طريق مراحل خطة المشروع على شكل مصفوفة تتبع (الشكل 14-4).



for Change of Value

Traceability for Change of Value (COV)

Feature	UC (UC extn)	UI	Sequence Diagrams	Project Plan
Change of Value (COV)	UC002 (UC002.2a)	X		X
COV.1) UI shows current value	UC002 (UC002.2a)	X		2.4.1 Display properties from a field object; 3.5.1 Asynchronous update
COV.1) a) MSLite subscribe to FSS' COV events	UC002 (UC002.2a)	X		2.1.1 Transmit COVS from FSS to PublishSubscribe
COV.1) b) MSLite receives FSS' COV event notification	UC002 (UC002.2a)	X		2.1.1 Transmit COVS from FSS to PublishSubscribe; 2.2.1 Publish subscribe bus
COV.1) c) UI receives notification of the COV & updates value	UC002 (UC002.2a)	X		3.5.1 Asynchronous update

for Issue Commands

Traceability for Issue Commands (IC)

Feature	UC (UC extn)	UI	Sequence Diagrams	Project Plan
Issue Commands (IC)	UC001 (UC001.3a; UC001.7a; UC001.10a)	1; 2; 3; 4; 5		X
IC.1) User issues commands from UI	UC001 (UC001.3a; UC001.7a; UC001.10a)	1; 2; 3; 4; 5		3.1.1 Capture command from User and transmit it to the AccessControl component
IC.2) UI sends command to MSLite	UC001 (UC001.3a; UC001.7a; UC001.10a)	1; 2; 3; 4; 5		3.1.1 Capture command from User and transmit it to the AccessControl component
IC.3) MSLite sends command to FSS	UC001 (UC001.3a; UC001.7a; UC001.10a)	1; 2; 3; 4; 5		3.2.1 Validate Command and relay to the Virtual FSS; 3.3.1 Virtual FSS to FSS
IC.4) UI reflect successful transmission of command to FSS	UC001 (UC001.3a; UC001.7a; UC001.10a)	1; 2; 3; 4; 5		3.3.1 Virtual FSS to FSS
IC.5) UI reflects failed transmission of command to FSS	UC001 (UC001.3a; UC001.7a; UC001.10a)	1; 2; 3; 4; 5		3.3.1 Virtual FSS to FSS

الشكل 14-4: القدرة على التتبع بدءاً من المتطلبات إلى خطة المشروع

14-3-4 الهيكلية والتصميم

لقد تم تحديد هيكلية النظام في مرحلة مبكرة لأنها المحور الذي يقوم عليه نجاح المشروع. يقوم الفريق المركزي بتوفير الأمور التالية للهيكلية التصميمية (Clements [et al.], 2002) في نظام Wiki:

- **عرض المكوّنات والارتباطات** (C&C): تبدأ مواصفات الهيكلية من المجالات الأكثر أهمية للنظام يليها واجهات التشغيل

للنظام. ويشمل هذا واجهات تشغيل المستوى الأول في النظام (المكوّنات والارتباطات)؛ وفي بعض المكوّنات الأكثر أهمية يتم إجراء مستوى ثانٍ من التجزيء لنفس الواجهة.

- **عرض العملية**: وهي تمثيل تشغيلي للعمليات التي لها علاقة في تنفيذ نظام (MSLite). تظهر المكوّنات التشغيلية الروابط بين واجهة المكوّنات والارتباطات من المستوى الأول مع العمليات، والآلية التي تتواصل بها العمليات المختلفة مع بعضها.
- **عرض الوحدة**: وهو عرض ثابت يُظهر وحدات تطبيق البرمجيات لنظام (MSLite) وتوابعه.
- **عروض تخصيص المهام**: تُظهر واجهة النشر مخططاً للأجزاء البرمجية لواجهات المكوّنات والارتباطات مع الأجزاء المتعلقة بالأجهزة، وتوضح الطريقة التي يتم بها توزيع الأنظمة الفرعية المختلفة للنظام (MSLite) مع مختلف المعدات. وتركز واجهة نظام الملفات على هيكلية المشروع المتعلقة بنظام الملفات التي سيستخدمها مبرمجو المشروع.
- **واجهات الاستخدام**: وهي عبارة عن واجهات للوحدات مرئية للعامة تصف بالتفصيل شروط المتغيرات والاستثناءات.
- **محددات السلوك**: توثّق هذه المحددات الطريقة التي تنفذ بها الهيكلية المفترضة للمتطلبات الوظيفية. تم استخدام المخططات المتسلسلة للإصدار 1,5 من لغة النمذجة الموحدة للتوثيق بناءً على واجهات الوحدات. والمقصود باستدعاء الوسائل الموضحة في المخطط تلك الوسائل الموجودة في الواجهات لوحدات خاصة بها.

من المهم ملاحظة أنه يتم إنجاز تصميم المتطلبات والهيكلية وإعادة عمل متعلق بها بتفصيل أكبر في مكان واحد في نظام Wiki.

14-3-5 أمور تقنية

نظراً إلى أهمية وجود بنية تحتية متكاملة جيداً في المواقع المختلفة الموزَّعة عبر الحدود الجغرافية، يقوم الفريق المركزي ببذل جهد لا بأس به في إنشاء بنية تحتية توفّر الحلول بشكل عملي. يتضمن نظام Wiki لمحة عن جميع الأدوات والتقنيات التي تتطلبها الفترة الكلية للمشروع. كما يقوم بتفصيل الإعدادات اللازمة لكل من المواقع التي تعمل في المشروع ويقوم بتفصيل استخدامات هذه الأدوات أيضاً.

إختار الفريق المركزي الحلول التي توظف أسلوب الاستعانة بالمصادر المفتوحة لمعظم الأدوات المستخدمة لأنها توفّر فوائد متعددة. وقد ظهرت هذه الحلول في مشاريع تطوير البرمجيات الموزَّعة المراكز بشكل أفضل من أدوات التجارة المتوفّرة؛ هذه الأدوات متوفّرة مجاناً، وهذه فائدة للفريق الذي يعمل عن بُعد لأنهم يعملون كمتدربين وهي متوفّرة بسهولة ولا تتطلب من الفريق المركزي أن يقوم بتخزينها أو اتباع أي إجراءات خاصة؛ كما تتوفّر فيها مجموعة كبيرة من الوثائق التي توضح طريقة إعدادها واستخدامها بشكل وافٍ.

حصل الفريق المركزي على مجموعة أدوات (borland) الخاصة بهندسة البرمجيات الحاسوبية، والتي تشتمل على أدوات تصميم وبرمجة (borland together) المتوافقة بشكل كبير من تطبيق (NET visual studio) وتطبيق (NET platform). سمحت (borland) للطلاب باستخدام مجموعة أدواتها مجاناً، بالإضافة إلى تدريب الفريق المركزي تقنياً، وتم تسجيل هذا التدريب بشكل مرئي وتوفيره عن طريق نظام Wiki لجميع المواقع التي تعمل عن بُعد.

كوّن الفريق المركزي مجموعة من المشرفين المدربين على استخدام نظام Wiki للأجزاء المتعلقة بالبنية التحتية. كما تم عقد

تدريبات مختصة بهذه الأجزاء؛ شمل هذا تدريبات تتعلق بتركيب الأدوات والتعامل مع أدوات إدارة الإعدادات واستخدام نظام Wiki أيضاً. يقوم جميع المبرمجين في الفريق الذي يعمل عن بُعد بإنشاء صفحات خاصة بهم وأخرى خاصة بالمجموعة في نظام Wiki، بحيث تعرض لهم خصائص نظام Wiki التي سوف يستدعونها عند الاستخدام خلال المشروع. قام الفريق المركزي بعمل محاضرات باستخدام العروض التقديمية فيها لعرض الأمور المهمة، كالتركيب واستخدام أجزاء البنية التحتية وإدارة المشروع والعمليات والأمور التي تم تعلمها من العمل في السنوات السابقة وقراءة مواصفات المتطلبات والهيكلية لنظام Wiki. وقد تم تصوير هذه المحاضرات أيضاً ووضعها في متناول اليد في نظام Wiki. فعندما تبدأ الفرق التي تعمل عن بُعد العمل، يتم تدريبها على عرض ومناقشة هذه التسجيلات كفريق واحد.

14-3-6 أمور استراتيجية: التخطيط والتحكم

لم تكن أمور التكليف بالعمل أو التخطيط للمشروع أو التحكم والمراقبة أموراً سهلة في أي وقت من الأوقات. بل وتصبح أيضاً أكثر تعقيداً عندما يكون هناك مجموعات موزَّعة على مناطق جغرافية مختلفة. ومن الأمور التي تجعل هذه الأمور صعبة اختلاف التوقيت الزمني الذي يعمل خلاله كل فريق واختلاف توقيت الالتزامات الأخرى خلال وقت العمل. إضافة إلى ذلك، فإن ما يعقِّد الأمور أكثر هو النقص في معرفة المهارات الفردية لكل عضو من أعضاء الفرق التي تعمل عن بُعد.

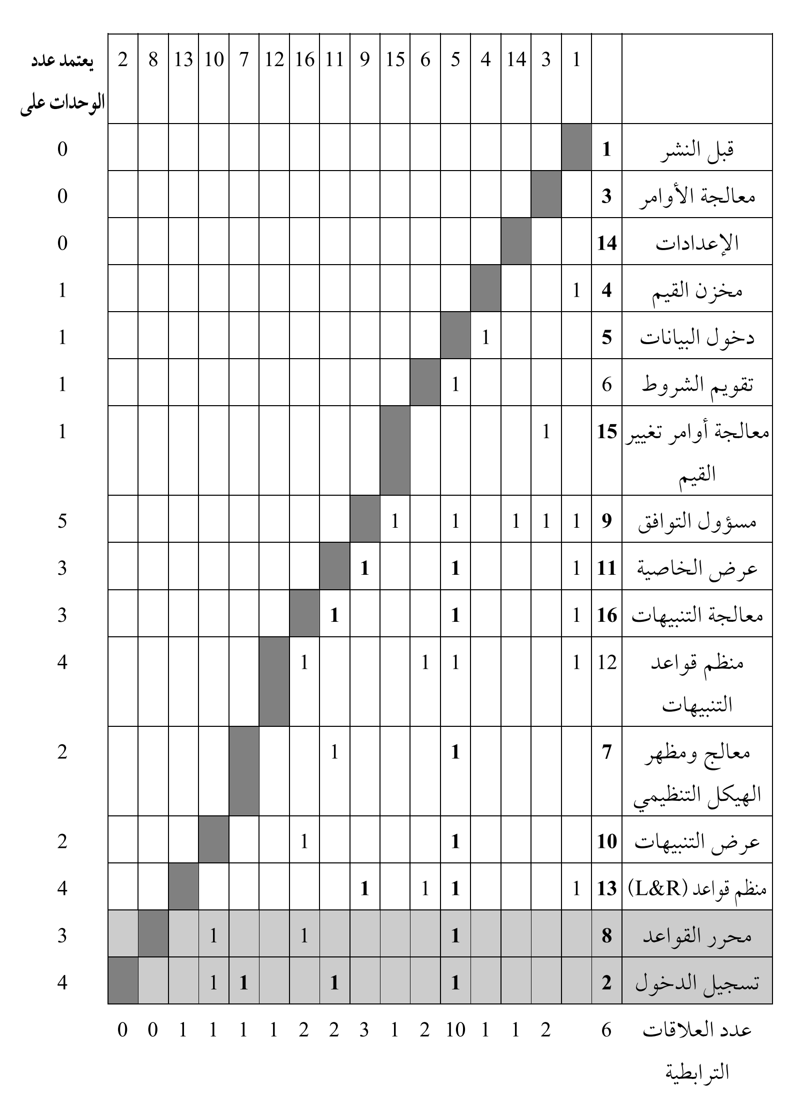
14-3-6-1 توزيع العمل

لدى الفريق المركزي خطوط عريضة يتم على أساسها التأكد من أنه قد تمت مواءمة عمليات توزيع العمل مع التعقيدات المتعلقة

بتطوير البرمجيات الموزَّعة المراكز. وهذه الخطوط العريضة هي:

- **أن تصبح الفرق خبيرة في مجال وظيفي محدد**: تم تصميم وإنشاء الوحدات بحيث تكون متماسكة بشكل كبير ولا تعتمد على بعضها بشكل كبير. وهذا يقلل الحاجة إلى تبادل التواصل بين الفرق التي تعمل عن بُعد، ذلك أن كلاًّ منها مسؤول عن وحدة معينة أو مجموعة وحدات لا تعتمد على الوحدات التي تقوم بإنجازها الفرق الأخرى.
- **إنشاء نظام** (MSLite) باستخدام طريقة البرمجة التزايدية: تنشأ كل فترة برمجية تكرارية عن إصدار جزء تنفيذي لقسم من نظام (MSLite). وهذا يعني أنه يتم إضافة الوظائف تدريجياً في كل فترة برمجية تكرارية.
- **إنشاء نظام** (MSLite) باستخدام طريقة البرمجة بالانتقال من الجزء إلى الكل: قبل برمجة كل وحدة، يجب أن تكون الوحدات التي تمثل متطلبات لوحدات أخرى قد تمت برمجتها أو في طور البرمجة. قام الفريق المركزي بإنشاء عرض لعلاقات الارتباط والجداول الزمنية المتعلقة بالنظام (سيتم الحديث عنها لاحقاً) لتحقيق هذا الهدف.
- **تجنب إنشاء الفرق التي يُطلب منها تعلم تقنيات متعددة**: أنشأ الفريق المركزي المنظور التقني للنظام وأوكل وحدات العمل إلى الفرق وفقاً للخبرات المتوفّرة لدى كل منها، لذلك يجب أن يكون المطلوب من كل فريق يعمل عن بُعد التركيز (وهذا يتطلب أن يكون متخصصاً) على تقنية معينة وليس العمل على مجموعة من التقنيات.

لقد تم استخدام التقنية الخاصة بمصفوفة هيكلية التصميم (DSM) (Browning, 2001) لفهم وعرض العلاقات الترابطية بين الوحدات الخاصة بالنظام (الشكل 14-5).



		1	3	14	4	5	6	15	9	11	16	12	7	10	13	8	2	يعتمد عدد الوحدات على
قبل النشر	**1**																	0
معالجة الأوامر	**3**																	0
الإعدادات	**14**																	0
مخزن القيم	**4**	1																1
دخول البيانات	**5**				1													1
تقويم الشروط	6					1												1
معالجة أوامر تغيير القيم	**15**		1															1
مسؤول التوافق	**9**	1	1	1		1		1										5
عرض الخاصية	**11**	1				**1**			**1**									3
معالجة التنبيهات	**16**	1				**1**				**1**								3
منظم قواعد التنبيهات	12	1				1	1				1							4
معالج ومظهر الهيكل التنظيمي	**7**					**1**				1								2
عرض التنبيهات	**10**					**1**					1							2
منظم قواعد (L&R)	**13**	1				**1**	1		**1**									4
محرر القواعد	**8**					**1**					1			1				3
تسجيل الدخول	**2**					**1**				**1**			**1**	1				4
عدد العلاقات الترابطية	6		2	1	1	10	2	1	3	2	2	1	1	1	1	0	0	

الترميز: 1 بالخط الغامق = تستخدم العلاقات الترابطية
1 بالخط العادي = علاقات ترابطية عادية

الشكل 14-5: مصفوفة التصميم الهيكلي لوحدات نظام (MSLite)

تساعد مصفوفة هيكلية التصميم في إنشاء عرض زمني لعمليات تطوير النظام، ويرمز هذا العرض إلى أبكر فترة زمنية يمكن فيها تطوير الوحدة مباشرة بعد أن تكون جميع الوحدات التي تلزم ذلك جاهزة، وأبعد فترة زمنية يمكن تطوير الوحدة فيها دون حدوث تأخير في الوحدات الأخرى التي تعتمد عليها وذلك إن وجدت.

تم إنشاء العرض التقني أيضاً، ما يوفر معلومات حول التقنيات التي يحتاجها تطبيق كل وحدة. والهدف من هذا العرض هو تحديد عدد التقنيات التي يحتاج فريق البرمجة إلى تعلمها لإنجاز وحدات العمل الخاصة به.

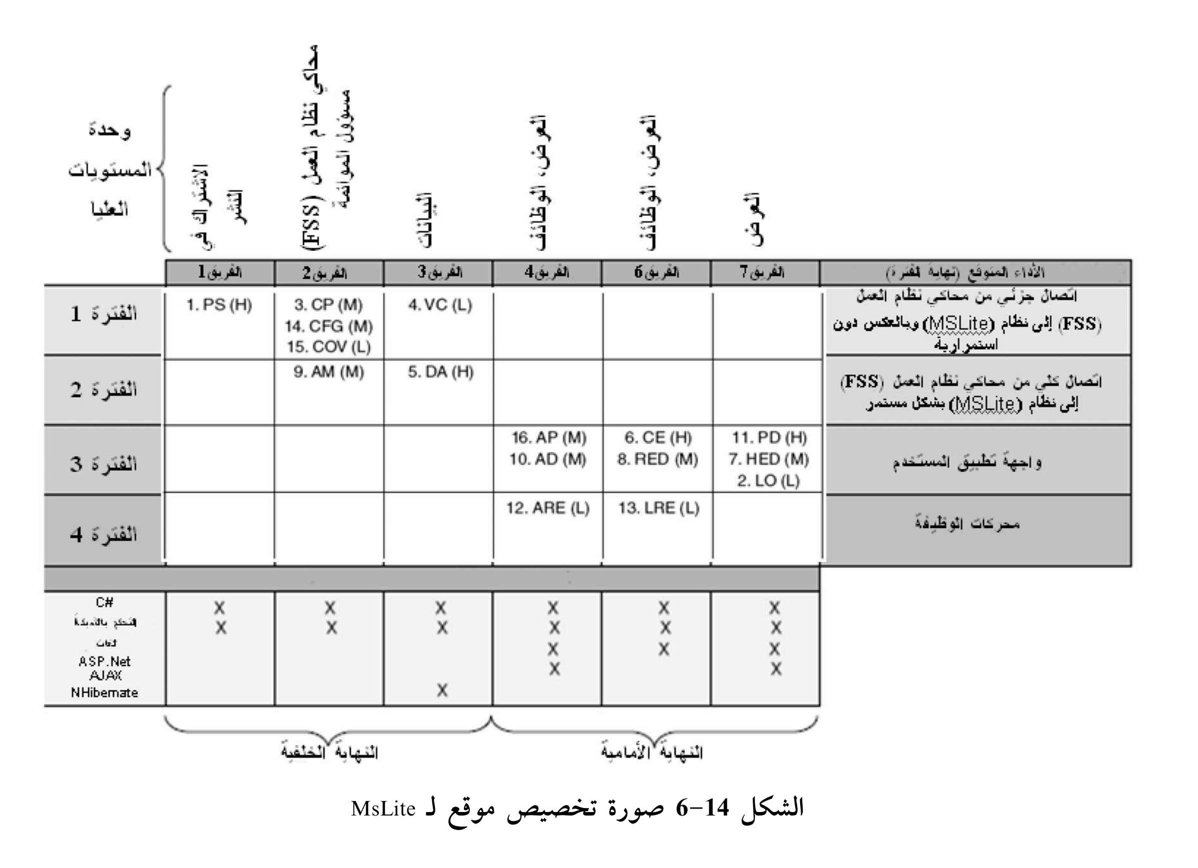
يستطيع الفريق المركزي إنشاء عرض مثالي لمهام العمل من خلال مجموعة من العروض لعمل البرمجة (الشكل 14-6). يحتوي هذا العرض على:

- الوحدات التي تم توكيل كل فريق من الفرق الست بها.
- الفترة الزمنية التي يجب خلالها تسليم كل وحدة.
- قدرة التشغيل المتوقعة في نهاية الفترة اعتماداً على الوحدات التي تم تنفيذها في هذه الفترة.
- التقنية التي يجب أن يعرفها كل فريق لتطبيق الوحدات الخاصة بهم.
- الوحدات المهمة في نظام (MSLite) التي سيكون كل فريق مسؤولاً عن تطبيقها.
- طول الفترة الزمنية التي يحتاجها الفريق في العمل في مشروع نظام (MSLite).
- الفترة الزمنية المتوفّرة للفرق لتعلم التقنيات التي يحتاجونها.

14-3-6-2 التخطيط للمشروع والتحكم به

يستخدم الفريق المركزي نظام Wiki للتخطيط للمشروع والتحكم به. يتم تفصيل المراحل التي تستمر خلال فترة تنفيذ مشروع التطوير بتوفير المعلومات الآتية في نظام Wiki:

- **الوظيفية**: القدرة التشغيلية الكلية المتوقعة في نهاية كل مرحلة.
- **الوظائف الأولية**: الوظائف الأولية التي تحتاجها كل وظيفة أساسية من أجل إنجازها.
- **المصادر**: الفرق التي تعمل على كل وظيفة أولية (هذا يأخذ بعين الاعتبار العروض المختلفة والمتعددة التي تم إنشاؤها للمهام الموكلة، والمهارات التي يمتلكها كل فرد من الفريق الذي يعمل عن بُعد في جميع الأدوات والتقنيات التي تم توزيعها كجزء من التدريبات التي يقوم به على نظام Wiki، وحالة ومستوى عمل كل عضو من أعضاء الفريق الذي يعمل عن بُعد).
- **موعد التسليم**: لكل وظيفة أولية.
- **وصف المسؤوليات**: تشمل وصفاً كبير الأهمية للمهام والمكوّنات والروابط والوحدات والواجهات والوسائل التي سيتم تطويرها، واختبار القبول للوظيفة الأولية، إضافة إلى أي معلومات أخرى متنوعة ذات علاقة حول التقنية التي تحتاجها الوظائف الأولية.



الشكل 14-6 صورة تخصيص موقع لـ MsLite

يعرض تقويم المشروع المعالم المهمة التي يجب إنجازها خلال مرحلة ما. وقد حدد الفريق المركزي أيضاً نموذجاً مخصصاً لتتبع عمليات المشروع والتحكم بها. يتم إنشاء هذا النموذج عن طريق مراحل العمل وعن طريق الوظائف الأولية أيضاً. ويُستخدم النموذج من كلا الفريقين، الفريق المركزي والفريق الذي يعمل عن بُعد لتحديد حالة مهام هندسة البرمجيات اللازمة لإكمال المهام المتعلقة بالوظائف الأولية. ويتطلب النموذج أيضاً أن تقوم الفرق بأخذ معايير الجودة (سيتم وصفها لاحقاً) ذات العلاقة بعملية تطبيق جميع الوظائف الأولية الموكلة إليهم بعين الاعتبار. ويُظهر النموذج أيضاً الوضع الحالي لعملية القبول واختبارات الدمج حالما تتم عمليات التسليم من الفريق الذي يعمل عن بُعد ليقوم الفريق المركزي بعمليات الدمج والاختبار. ولا يتم تحديد الحالة من مؤشرات مسبقة لانتهاء المهمة المعنية، ولكن ما إنْ يتم ربطها بالأعمال المتعلقة بها. حتى يتم إنشاء معظم هذه المهام باستخدام نظام Wiki ويتم تنظيمها باستخدام وحدات ذات مستوى متقدم. وهناك نموذج يُستخدم في تحديد مخططات الطبقة لكل وحدة عالية الأهمية، ومخططات التتابع، وتحديد حالة اختبار الوحدة، ومراجعة النص البرمجي، وغير ذلك.

مع وجود برنامج زمني للاجتماعات بين كل فريق يعمل عن بُعد ومدير التزويد المسؤول عنه في الفريق المركزي، يتم متابعة حالة إنهاء العمل في المهام ومدى جودتها بشكل مستمر.

14-3-7 ضمان الجودة

يوجد نواحٍ متعددة لضمان الجودة تم إدراجها في الممارسات والعمليات التي تجري على المشروع:

- **معايير الجودة**: هذه المعايير التي حددها الفريق المركزي لما يسلمه الفريق الذي يعمل عن بُعد. وقد تم تصنيفها إلى معايير الأخطاء، ومعايير عمليات المراجعة وإبداء الرأي، ومعايير الإجراء، والمعايير للمشروع ككل. وباستخدام ما مجموعه 14 معياراً موزّعاً على هذه الجوانب الأربعة، ويقوم مدير التزويد أسبوعياً بتقويم ما تسلمه الفرق، وعن طريق استخدام عملية تقويم الفريق التي تم ذكرها سابقاً، يتم إعلام أعلى فريقين من حيث التقويم كل شهر، ويتم مكافأتهما.
- **معايير لغة البرمجة**: قام الفريق المركزي بسرد معايير لغة البرمجة (C#) التي تُعتبر أساساً للخطوط العريضة للغة البرمجة المستخدمة في برمجة الأعمال المقترحة. ومع ذلك، فبما إن أحد معايير الجودة التي يتم تقويم الفريق الذي يعمل عن بُعد على أساسها هي عدد التجاوزات لمعايير لغة البرمجة، وعلى هذا الأساس يُتوقع من أفراد الفريق أن يلتزموا بجميع معايير لغة البرمجة، أو يقوموا بتطوير هذه المعايير عن طريق عملية إدارة التعديل وبإشراف مدير التزويد المسؤول عنهم.
- **اختبارات القبول**: يتم سرد اختبارات القبول لكل وظيفة فرعية عن طريق مدير ضمان الجودة في الفريق المركزي. ويتم توفير هذه الاختبارات لجميع الوظائف الفرعية التي تشكل مجتمعة مرحلة عمل قبل أن يتم إنجاز هذه المرحلة.
- **اختبارات الدمج**: كما هو الأمر في اختبارات القبول، يتم أيضاً تحديد اختبارات الدمج والحفاظ عليها عن طريق مدير ضمان الجودة لكل وظيفة في مرحلة العمل وقبل المباشرة في عمليات الدمج لمرحلة عمل معينة.
- **التحقق من شيفرة البرمجة واختبار صحتها وفعاليتها ومراجعة النص البرمجي**: يتم اختبار شيفرة البرمجة ومراجعة الشيفرة المصدرية وصيانتها بواسطة الفريق الذي يعمل عن بُعد. وعن

طريق استخدام النموذج (يتم إنشاؤه عن طريق وحدات عالية الأهمية) الذي تم تزويد الفريق الذي يعمل عن بُعد به في نظام Wiki، يقوم الفريق الذي يعمل عن بُعد بتوثيق وضع هذه النواحي، وإنشاء روابط مع طبقات اختبار الوحدة التلقائي في المستودع البرمجي.

14-3-8 التدريب

إن تدريب الأشخاص على العمليات والأدوات والتقنيات وعلى المشروع بشكل عام هو أحد المعايير المهمة للنجاح. وتتناسب مقدرة أي تنظيم على أداء ذلك بكفاءة وفاعلية عكسياً مع درجة التشتت الجغرافي لأعضاء الفريق. ويستخدم الفريق المركزي عدداً من التقنيات المختلفة في السنة الثانية للمشروع بعد إدراك مغزى ومدى صعوبة الوصول إلى هذا الهدف. ويتم تحديد التفاصيل لكل ناحية من النواحي بدرجات متفاوتة من التفصيل في نظام Wiki باستخدام عرض متلفز في معظم الحالات. ويتم إنشاء حسابات مستخدمين لأعضاء الفريق الذي يعمل عن بُعد على الأقل قبل أسبوع من الاجتماع الأول، ويُطلب منهم أن يبدأوا القراءة من خلال نظام Wiki. يكون الاجتماع الأولي عادة مع الفريق الذي يعمل عن بُعد عن طريق شبكة متلفزة، ويوجد أيضاً قائمة مفصلة بالمهام يعمل مدير التزويد على أساسها بإجراء جولة تعليمية للفريق الذي يعمل عن بُعد حول نظام Wiki.

يُمضي كل فريق يعمل عن بُعد عدة أسابيع في دراسة التمارين التي تم إنشاؤها خصيصاً لهم لزيادة اضطلاعهم بالبنية التحتية والأدوات والعمليات. ومن ثم ينتقلون إلى زيادة اضطلاعهم حول المتطلبات والهيكلية وخطة المشروع. وأما بخصوص الفرق التي

تعمل بالفعل، فتتم هذه العملية على الأكثر قبل شهر من استعدادهم وجهوزيتهم للبرمجة الفعالة. خلال هذا الشهر، يقوم مدير التزويد بمساعدة الفرق التي تعمل عن بُعد في دعم استيعابهم للعمليات والنواحي الأخرى للمشروع. حتى هذه اللحظة، لم يواجه الفريق المركزي أي حاجة إلى إعادة تدريب أي من الفرق على أي من هذه النواحي، ما لم يحدث هناك تغيير كبير على أي من هذه النواحي. في هذه الحالة، يقوم الفريق المركزي بتخصيص جزء من وقت أحد الاجتماعات الأسبوعية لتوزيع المعلومات الجديدة والمحدثة المتوفّرة عن نظام Wiki.

14-4 الوضع الحالي للجهود المبذولة في تطوير نظام (MSLite)

في الوقت الحالي (في الأول من كانون الأول/ ديسمبر 2005) تم وضع الأساس لمتطلبات مشروع نظام (MSLite). تتوفّر في نظام Wiki كافة المكوّنات والروابط والوحدات وعملية النشر وتوزيع الهيكلية ونموذج المجال كبير الأهمية للنظام ومواصفات واجهات التطبيق (الواجهات المرئية للمكوّنات المعروضة للعامة) والمواصفات الأدائية (مخططات متتابعة خلال الواجهات المرئية المعروضة للعامة). لقد تم إنشاء هذه الأمور من خلال خطة مشروع متماسكة ومسيطر عليها. وقد باشرت فرق العمل في الجامعة البابوية الكاثوليكية في ريو غراند دو سول، وجامعة كارنيغي ميلون، وجامعة مونموث، العمل في عمليات البرمجة للمرحلة الأولى. أما فريق جامعة ليميريك فهو بصدد البدء، وهو الآن في مرحلة مدتها شهر واحد من التدريب. ويُتوقع أن يباشر فريق جامعة ليميريك في العمليات البرمجية منذ بداية المرحلة الثانية.

14-5 الخطوات التالية لمشروع نظام (MSLite)

بالنظر إلى الهدف الأكبر وهو تنفيذ المشاريع الموزَّعة جغرافياً بطريقة فعالة ومستمرة، تم على هذا الأساس تحديد خطوات مستقبلية ملموسة، وهي:

- إدراج عملية تقريبية وآلية ليصبح من الممكن توقع المراحل المستقبلية والتخطيط لها.
- إعداد نظام لتتبع المشاكل وتحديد العمليات اللازمة لذلك، وتحسين عملية إدارة التغيير.
- فهم وتوثيق أهمية نظام Wiki في نجاح الجهود المبذولة في مشاريع تطوير البرمجيات الموزَّعة المراكز، وكيف يمكن إجراء تحسينات على استخدامه، وكيف يمكن الارتقاء به من أجل استخدامها بفعالية وكفاءة في المشاريع التجارية لتطوير البرمجيات.
- جمع المعلومات عن طريق شبكة الإنترنت، وتحليل الشبكات الاجتماعية والأدوات التي تعمل تلقائياً، وإجراء التحليلات لتحديد الأمور التي يجب التركيز عليها، وكذلك تحليل الأنماط التجارية التي قد تكون ذات أهمية من أجل فهم وتنفيذ ناجح لمشاريع تطوير البرمجيات الموزَّعة المراكز.
- فهم وتنظيم الطريقة التي تؤثر بها الهيكلية وما يتعلق بها على الشركة، وبالعكس.

المراجع

Books

Clements, Paul [et al.]. *Documenting Software Architectures.* Boston, MA: Addison-Wesley, 2002.

Hofstede, Geert. *Cultures and Organizations: Software of the Mind*

- *Intercultural Cooperation and Its Importance for Survival*. Revised Edition. New York: McGraw-Hill, 1997.

Periodicals

Browning, Tyson R. «Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Direction.» *IEEE Transactionson Engineering Management*: vol. 48, no. 3, 2001, pp. 292-306.

Herbsleb, James D. and Deependra Moitra. «Global Software Development.» *IEEE Software*: vol. 18, no. 2, 16-20 March/ April 2001.

Perry, D. E., N. A. Staudenmayer, and L. G. Votta. «People, Organizations, and Process Improvement.» *IEEE Software*: vol. 11, no. 4, 1994, pp. 36-45.

Conferences

Herbsleb, James D., Daniel J. Paulish and Matthew Bass. «Global Software Development at Siemens: Experience from Nine Projects.» *Proceedings of the 27th International Conference on Software Engineering*, St. Louis, MO., 2005, pp. 524-533.

website

<http:// meta. wikimedia. org>.

الفصـــل الخامس عشر

نظام معالجة البيانات

15-1 تمهيد

نظام معالجة البيانات DPS2000 هو نظام برمجي يستخدم للحصول على القراءات وتحليلها في بعض الأجهزة كأجهزة قياس استهلاك الكهرباء والغاز والماء. وقد تم إنشاء مشروع نظام معالجة البيانات DPS2000 في العام 1999. وتم توثيق المشروع في السابق كدراسة حالة في شركة سيمنز (Paulish, 2002). وتمت عملية البرمجة في أربعة مواقع تابعة لشركة سيمنز في ثلاث دول هي سويسرا وألمانيا والولايات المتحدة الأميركية. ويتم حالياً تسويق وتوزيع المنتج بنجاح. يمكن اعتبار مشروع نظام معالجة البيانات DPS2000 مشروعاً ناجحاً وفق المعايير المتعارف عليها الخاصة بالموازنة والبرنامج الزمني ومدى الجودة والأهداف الوظيفية.

تم برمجة المنتج بشكل مبدئي باستخدام خمس مجموعات من التطبيقات التي تم إجراء التخطيط لها بحيث يتم تطبيقها على

مكوّنات نظام معالجة البيانات DPS2000. وتم تشكيل فريق تصميم هيكلي، يتكون من خمسة مهندسين وخليط من خبراء في هذا المجال وفي التصميم الهيكلي. وتم تعيين مسؤولٍ عن التصميم وبدأ الفريق في التصميم الهيكلي بتحليل احتياجات السوق، وإنشاء الشجرة المفاهيمية، وتقصي تقنيات التطوير التي يمكن استخدامها، وإنشاء النماذج. يتكوّن الفريق المركزي من أعضاء من عدة مواقع تطوير عملوا في مراحل مبكرة من المشروع وبشكل رئيس في الموقع المركزي. وحالما بدأت العمليات البرمجية، عمل المصممون بشكل أساسي في مواقعهم الأصلية، مع وجود اجتماعات متكررة في أحد المواقع تم وضع خطة زمنية لها ليتم دمج الأنظمة البرمجية الفرعية عندما تكون جاهزة.

15-2 التحليل الشامل

تم إجراء التحليل الشامل على نظام معالجة البيانات DPS2000 كإحدى المهام لفريق التصميم الهيكلي. كان أحد أهم العوامل التنظيمية المؤثرة التي ظهرت خلال التحليل الشامل هي عامل المهارات التقنية اللازمة لتنفيذ حزم التطبيق، فقد ظهر هناك نقص في هذه المهارات في التنظيم المركزي لأن المنتجات السابقة كانت تعتمد على نظام UNIX ومنشأ على أساس واجهات مستخدم محلية ولكن نظام معالجة البيانات DPS2000 اعتمد على نظام وندوز (Windows) ومنشأ على أساس واجهات مستخدم متشعبة. وكما يحدث عادة في شركة سيمنز، كان التنظيم المركزي عبارة عن شركة ربحية يوجد فيها عدد محدود من فرق العمل التقنية، أقل من اللازم لتطوير منتج بحجم نظام معالجة البيانات DPS2000 وخلال الفترة الزمنية المطلوبة. لذلك تم التخطيط للتطوير الشامل للتعامل مع هذا العامل المؤثر، بحيث يتم توفير المهارات المطلوبة في ثلاثة مواقع

تطوير إضافية تابعة لشركة سيمنز تعمل عن بُعد عن الموقع المركزي. وأيضاً تم تطوير مستوى إضافي لوثائق مواصفات التصميم الهيكلي بمستوى أقل من التصميم كبير الأهمية. يقوم هذا النظام بتصميم المواصفات عن طريق التركيز على وصف الروابط والعلاقات بين الأنظمة الفرعية الرئيسة للبنية الهيكلية، وتم توزيع هذه الأنظمة الفرعية في مواقع البرمجة التي تعمل عن بُعد.

أحد العوامل التنظيمية الأخرى هو أن الإدارة أرادت عرض المنتج في السوق بأقصى سرعة ممكنة، لأن السوق كانت في حالة تغير سريعة، وكان يؤمل أن يتم تسليم المنتج بوظائف محدودة إلى المستخدمين المحتملين من أجل جمع تغذية مرتجعة وآراء المستخدمين حول المنتج. فقد كانت الاستراتيجية المتبعة للتعامل مع هذا العامل هي تطوير المنتج بشكل مرحلي باستخدام أسلوب التجزيء الزمني بحيث يتم الالتزام بمواعيد الجدول الزمني حتى وإن افتقد الإصدار لبعض الخواص. وقد استخدم نظام معالجة البيانات DPS2000 من ست إلى ثماني مراحل تطوير أسبوعية لكل إصدار هندسي. هذه الفترة الزمنية للمراحل مكّنت فريق البرمجة من إنتاج مجموعة جيدة من الخصائص التي يمكن اختبارها وتقويمها من قبل فريق الاختبار.

تم وصف الكثير من الممارسات المستخدمة في مشروع نظام معالجة البيانات DPS2000 في هذا الكتاب، وكان ذلك المشروع ناجحاً، وبالتالي أصبح من المراجع الدراسية لـ «الممارسات الأفضل». ومن التعديلات المهمة التي تم إجراؤها منذ تنفيذ هذا المشروع هي إنقاص الفترة اللازمة للمراحل من ستة أو ثمانية أسابيع إلى فترة شهر واحد تم التخطيط لها بحيث يتم العمل بشكل متزايد. وقد تم ملاحظة أن الإصدارات الشهرية للمهام البرمجية جعل المشروع يسير

بشكل منتظم وعرض رؤية أفضل للتقدم في سير المشروع بحث يتم مراجعة ذلك في كل شهر. ولا نعتقد أن الفترة الزمنية الشهرية متقاربة أكثر من اللازم لتناسب مشاريع التطوير الموزَّعة لأن مرحلة التطوير تتكون من عدة فترات زمنية يستمر العمل فيها لمدة سنة أو أقل.

15-3 استراتيجيات التصميم

تحدد الاستراتيجيات المتبعة في عملية التصميم خواص الهيكلية والقيود المفروضة عليها وتساعد في تحديد المخاطر المحتملة المتعلقة بتطبيق النظام البرمجي. وكنتيجة للتحليل الشامل لنظام معالجة البيانات DPS2000، تم تحديد أربع وعشرين استراتيجية تصميمية موجهة للتعامل مع العوامل المؤثرة. وتم استنتاج ستة أمور من هذه الاستراتيجيات الأربع والعشرين، وتم استخدامها كمبادئ توجيهية للمشروع. وأحد المبادئ التوجيهية الستة هو أن تتم البرمجة في عدة مواقع. ونعرض هذا المبدأ مرة أخرى كما باوليش (Paulish, 2002) في ما يأتي.

● التطوير في عدة مواقع: وجود نقص في الكفاءات التقنية المناسبة في موقع واحد كان من العوامل المؤثرة التي تم التعامل معها عن طريق إنشاء أربعة مواقع مختلفة موزَّعة في ثلاث دول للعمليات البرمجية. ويضع هذا القيود على عملية التصميم لأن توزيع المهام يكون أكثر صعوبة حين وجود عدة مواقع، ويجب أيضاً إعداد البيئة والأدوات بحيث تناسب وجود مواقع عمل متعددة.

15-4 هيكلية نظام معالجة البيانات

تم تصميم الهيكلية لنظام معالجة البيانات DPS2000 بحيث يمكن إضافة الأنظمة الفرعية إلى الطبقة المنطقية للعمل. وإضافة إلى

ذلك، تم اكتساب البيانات بواسطة نظام جزئي تم الحصول عليه من منتج آخر لشركة سيمنز ومن ثم تم تعديله. تقوم الأنظمة الفرعية بعملية تبادل المعلومات خلال الجداول في قاعدة البيانات المتداولة. الشكل 15 ـ 1 يوضح ملخصاً للبنية الهيكلية في نظام معالجة البيانات DPS2000.

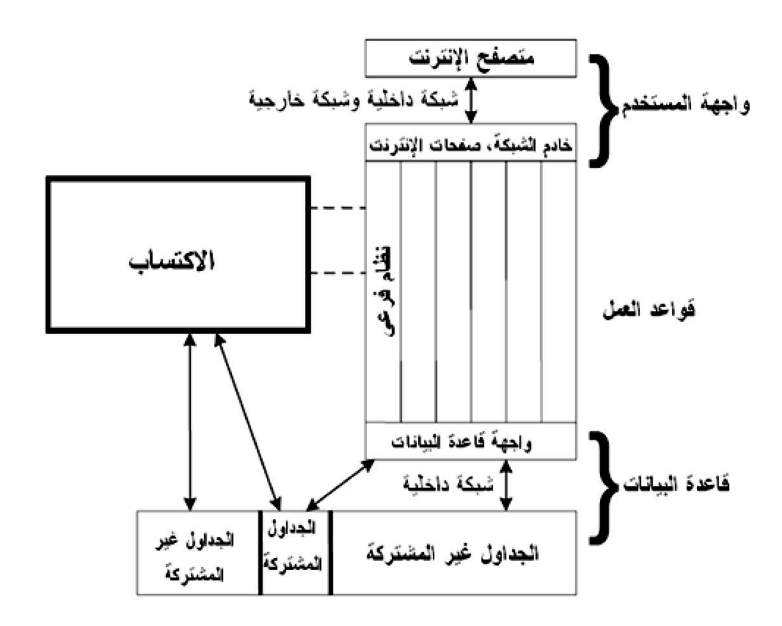
15-5 التخطيط للمشروع

تم التخطيط لمشروع تطوير البرمجيات في نظام معالجة البيانات DPS2000 كمجموعة متتابعة من إصدارات هندسية تزداد وظائفها بشكل تدريجي. وقد تم إنشاء الجدول الزمني للمشروع بحيث يتم العمل في الإصدار التالي، بينما يقوم المستخدمون وفريق الاختبار الذين يعملون من المنزل باختبار الإصدار الحالي. وقد تم إنشاء الخطط الخاصة بالإصدارات أيضاً بشكل يتوافق مع مواعيد معارض التجارة الصناعية، بحيث يجب طرح إصدارات جديدة بأحدث مواصفات ممكنة. وفي عملية تخطيط مشروع نظام معالجة البيانات DPS2000، تم تحديد وقت ثابت للإصدارات ومراحل للعمل، ومن ثم تم تخطيط وظائف كل إصدار في كل مرحلة عمل اعتماداً على التقديرات المتزايدة ومتطلبات الاختبار.

تتناسب عملية البرمجة المتكررة بشكل جيد مع مشاريع التطوير الموزَّعة. وقد وجدنا أن استخدام النظام نفسه هو من أفضل الوسائل الخاصة بالتواصل بين الفرق الموزَّعة في المواقع الأربعة. على سبيل المثال، بعد إصدار نماذج أولية لأجزاء من نظام معالجة البيانات DPS2000 أصبح من الممكن تحقيق فهم كامل ومناقشة المتطلبات فقط. وقد يكون هذا لأن عمل النظام نفسه يصبح موضوعاً مشتركاً للجميع. وتقوم البرمجة المتكررة أيضاً بتشجيع المبرمجين على التعلم

السريع والبدء باستخدام تقنيات جديدة لأنهم يكونون حينها بحاجة إلى التعلم بشكل أكبر ليصبحوا قادرين على التطبيق، حتى بالنسبة إلى الوظائف الجزئية.

المعالجة



الشكل 15-1: هيكلية ذات ثلاث مستويات لنظام معالجة البيانات 2000
(DPS2000)، **من:** (Daniel J. Paulish, *Architecture-Centric Software Project Management* (Boston: Addison-Wesley, 2002). مع أخذ الموافقات اللازمة.

15-6 إدارة المشروع

يعمل في كل موقع من المواقع الأربعة الخاصة بتطوير نظام معالجة البيانات DPS2000 مدير محلي للمصادر للقيام بالأعمال

الإدارية لأعضاء الفريق. ويوجد أيضاً مدير عام لمشروع نظام معالجة البيانات DPS2000، وأيضاً مديرو مشاريع لكل عملية تطوير لمجموعة من مجموعات التطبيقات البرمجية. وبذلك، يكون هناك مديرون على المواقع المحلية ومديرون عامون للمشروع مسؤولون عن تحقيق أهداف المشروع. فعلى مديري المشروع مناقشة مهام العمل الفردية؛ على سبيل المثال، إذا كان مخططاً أن يعمل أحد الأفراد على برمجة عدة مجموعات من التطبيقات البرمجية في نفس الوقت، يقوم المدير العام للمشروع بإيجاد حلول للأمور التي تتعارض مع بعضها بعضاً.

لقد كان رئيس قسم التصميم الهيكلي مسؤولاً عن أخذ القرارات وإيجاد الحلول للأمور المتعارضة تقنياً لمجموعات التطبيقات. وكانت هذه الوظيفة تعتبر مهمة في المشروع وتمتلك صلاحيات شاملة. في الواقع العملي، تمت مراجعة القرارات التقنية المهمة التي أثرت في أهداف المشروع من قبل كل من مديري المشروع والمديرين التقنيين قبل أن يتم تطبيقها.

إن كل نظام فرعي تم تصميمه وتطبيقه في نظام معالجة البيانات DPS2000 تم وضع أحد المهندسين ليكون مسؤولاً عنه. إضافة إلى ذلك، تم جمع المصادر اللازمة لتطبيق أحد الأنظمة الفرعية في موقع واحد.

تم تتبع حالة العمل في المشروع خلال الاجتماعات المتلفزة الأسبوعية. وتم حث كل عضو من أعضاء الفريق أن يقدم تقريراً حول سير العمل البرمجي، وأن يقوم بعرض أي معلومات أو أمور أخرى لمشاركتها مع الأعضاء الآخرين.

يعمل في مشروع نظام معالجة البيانات DPS2000 تقريباً عشرون

مهندس برمجيات تم تعيينهم خلال ذروة العمليات البرمجية؛ لذلك كان من الممكن أن تقوم الفرق بتقديم تقارير أسبوعية حول سير العمل البرمجي خلال الاجتماعات الأسبوعية المتلفزة.

15-7 دروس مستفادة

تعامل بعض أعضاء فريق مشروع نظام معالجة البيانات DPS2000 مع عملية تطوير مواصفات تصميم النظام على أنها غير ضرورية، وتأخر بدء العمل في تطبيق المجموعات البرمجية. ولكن تمكَّن أعضاء فريق برمجي جدد يعملون على كل مجموعات التطبيقات القديمة والجديدة من استخدام هذه المواصفات بشكل ناجح. وكان تجزيء رزم العمل على المواقع الأربعة نقطة مهمة وحرجة. وقد لاحظنا أن عملية الدمج للأنظمة الفرعية المختلفة تمت بسلاسة مطلقة حين وجود قادة هذه الأنظمة الفرعية في موقع واحد.

تكون لدينا خبرة جيدة في أسلوب البرمجة التزايدية في مشروع نظام معالجة البيانات DPS2000. ومن خلال نشر العنوان الإلكتروني للموقع الذي تتم فيه عملية اختبار النظام التي تتم فعلياً في سويسرا، استطاع جميع أعضاء الفريق ومديريهم من مراقبة تقدم العمل البرمجي كلما تمت إضافة خاصة جديدة. وقد كان هذا عاملاً مهماً في رفع معنويات الفريق، إذ أصبح كل عضو في الفريق مدركاً للتقدم السريع الذي تم إنجازه بعد اكتمال مرحلة التصميم كبيرة الأهمية، وتمت مباشرة العمليات البرمجية. وقد كانت أول الإصدارات الهندسية تطبيق شريحة عامودية في الهيكلية. وقد ساعد هذا في التأكد من صحة الهيكلية ومنح الفريق البرمجي الثقة وزاد فهمهم للبنية الهيكلية بحيث يصبحون قادرين على تطبيق إصدارات هندسية أخرى في المستقبل.

بما إننا وضعنا أولويات في الجدول الزمني بخصوص

الإصدارات وعمليات تبادل الوظائف كلما دعت الحاجة، استطاع الفريق البرمجي أن ينجز الإصدارات في موعدها المحدد. وقد ساعد هذا في تكوّن صورة لصدقية الفريق البرمجي من قبل المديرين لأنهم كانوا على علم بأن مجموعة جديدة من الوظائف سوف تكون جاهزة لتتم عليها عمليات التحقق في التواريخ المخطط لها في بداية العمل. لحسن الحظ، تمت المحافظة على جودة عالية للبرمجيات وثبات جيد للنظام خلال العمليات البرمجية، ولذلك لم يكن صعباً التوفيق بين المحافظة على المواعيد وجودة البرمجيات.

في بداية المشروع، تم عقد اجتماعات شهرية للمشروع، بشكل متناوب في مواقع البرمجة المختلفة. وفي وقت لاحق، تم عقد الاجتماعات بشكل رئيس عند دمج الأنظمة الجزئية الرئيسة أو عند وجود حاجة للتدريب أو حاجة لتوضيح التفصيلات حول التصميم لمهام برمجية جديدة. عقدنا اجتماعات متلفزة أسبوعية للوقوف على وضع الجدول الزمني وتوضيح بعض المشاكل العامة التي يجب على المبرمجين أن يكونوا مدركين لها. وكإجراء إضافي، قمنا بنشر أهداف الاجتماع المتلفز، باستخدام التقنيات التي تعلمناها خلال فترة تدريب الفريق وإنشائه للتخطيط للاجتماعات وعقدها.

تم عقد ورشتي عمل لجميع الفرق في مرحلة مبكرة للمشروع، بحيث شعر الجميع بقيمة المشروع. الورشة **الأولى** كانت تختص في عملية بناء الفريق تم عقدها في أحد المواقع البعيدة في سويسرا. بالإضافة إلى الإجراءات الاعتيادية لبناء الفريق، تم التعامل مع مهام محددة، مثل تطوير عملية قياسية للمشروع. وشملت الورشة تخصيصاً لبعض الوقت من أجل أن يتعرف أفراد الفريق على بعضهم بعضاً ويتبادلون النقاشات على شكل أزواج حول مشاركتهم واختلاف الآراء حول ذلك.

الورشة **الثانية** كانت ورشة متعددة الثقافات عُقدت في مبنى

شركة سيمنز في برينستون، نيوجيرسي. وكان راعي هذه الورشة مدرباً على الثقافات المختلفة في سويسرا وألمانيا والولايات المتحدة الأميركية. وكان رد الفعل على هذه الورشة إيجابياً أيضاً، لأنه من الواضح أن صبر أعضاء المجموعة كان يزداد كلما زاد تعلمهم لثقافات الدول الأخرى. وكان أعضاء الفريق البرمجي قد نشأوا وترعرعوا في عشر دول مختلفة، ولكن الورشة ركزت بشكل أساسي على الدول الثلاث التي تتمركز فيها مواقع البرمجة. وكان أحد نتائج هذه الورشة التركيز على تحديد المهام والوظائف لأفراد الفريق. وكان هذا نتيجة إدراك أن هناك ثقافات معينة تثمن العمليات والوظائف المحددة بشكل جيد، وكذلك المواقع ووضع كل عضو من أعضاء الفريق في التنظيم المركزي. ويدعم عقد مثل هذه الاجتماعات فكرتنا بأنه يجب تطوير علاقات العمل الشخصية بين الأعضاء المهمين للفريق المركزي والمواقع التي تعمل عن بُعد لتسهيل عملية التواصل حين ظهور أي عائق في عملية التطوير.

يُمضي أعضاء فريق البرمجة جزءاً كبيراً من وقتهم في عمليات التواصل والتفاعل أثناء الاجتماعات؛ لذا، لا بد من فهم الأنماط الثقافية المختلفة التي يتم استخدامها. ويميل بعض أعضاء الفريق إلى تقديم تقرير متفائل عن تقدم العمل. لكن قد يشعر بعضهم بعدم الراحة أو بتخوف بسبب مناقشة وسائل تقنية مختلفة من قبل المؤيدين لكل وسيلة. وقد شارك بعضهم في عملية التواصل وبعضهم لم يفعل. وما كان مهما بالنسبة إلى الإداريين، وما تم مراجعته بالنسبة إلى فريق البرمجة هو طرح منتج عالي الجودة في الأسواق وفي الوقت المحدد ضمن الموازنة الموضوعة. لذلك، يقوم منسقو الاجتماعات أو مديرو الفريق، الذين أدركوا الاختلافات الثقافية وكانوا قادرين على تحفيز نتائج أفضل لكل فرد من المهندسين، بمساهمة جيدة للمشروع.

قد تؤثر الاختلافات الثقافية في القيم مثل الالتزام بالمواعيد والإتقان وأخلاقيات العمل والعمل الجماعي والجودة والتفاعل في القرارات المتعلقة بالمشروع. وقد تحسن هذه الاختلافات عملية سير المشروع وقد تعيقها. وبالطبع، تُعتبر الإجازات من إحدى أهم المشاكل المتعلقة بالفروقات الثقافية التي تواجه مدير المشروع. ومن الصعب جعل البرنامج الزمني للمشروع متوافقاً بين الفريق المركزي والفريق الذي يعمل عن بُعد، بخاصة إذا طرأت عطل محلية بشكل مفاجئ في المواقع الأخرى. وتشكل الفروقات في التوقيت مشكلة في التواصل خلال العمل اليومي. وفي حالتنا هذه، هناك فارق مقداره ست ساعات في التوقيت.

فكرة مفيدة:

شارك في برامج السفر دائماً.

إذا تم تعيينك في أحد مشاريع التطوير الموزَّعة، فلا بد من أنك ستسافر إلى بعض المواقع التي تعمل عن بُعد. إن السفر بهدف التواصل وجهاً لوجه أمر ضروري لنجاح مشاريع التطوير الموزع، ومن أجل تشجيع مديري المشاريع لتخصيص موازنة مناسبة للسفر. ومع ذلك، قد تعاني في كثير من الأحيان من اضطراب نتيجة الرحلات الجوية، وبالتالي لن تعمل بكفاءة عالية عندما توجَد في المواقع التي تعمل عن بُعد. لذا، شارك في برامج السفر دائماً وافعل ما بوسعك لتكون هذه الرحلات مريحة لتحافظ على كفاءتك في العمل بعد الوصول.

15-8 الملخص

على الرغم من الجهود الحثيثة التي بُذلت للتواصل بين المواقع

الأربعة والتركيز على الوثائق الخاصة بالتصميم مع وجود واجهات معدة بشكل جيد، وجدنا أن التطوير الموزع كان بشكل واضح أكثر صعوبة من التطوير في موقع واحد. وكان هذا نتيجة وجود خلل في عملية التواصل من وقت إلى آخر وذلك بسبب اختلاف في مواعيد الإجازات والعطل الرسمية من دولة إلى أخرى، واختلاف في التوقيت، وأيضاً بسبب أعطال في الحواسيب أو الشبكات في بعض الأحيان. على سبيل المثال، إذا تم توجيه سؤال إلى الزملاء الموجودين في أوروبا خلال ساعات المساء من قبل الفريق الذي يعمل في الولايات المتحدة خلال ساعات عملهم، يجب عليهم الانتظار حتى اليوم التالي للحصول على الإجابة. ولتدارك الأمور غير المتوقعة، يقوم أعضاء الفريق عادة بالتواصل مع زملائهم في الفرق الموجودة في دول أخرى عن طريق هاتف المنزل، وتمت إعادة برمجة النظام كل يوم تقريباً في عدة مواقع باستخدام أحدث شيفرة متوفّرة. كما تم الاهتمام بالتدريب المتعلق بالتقنيات وبناء الفريق والتدريب المتعلق بتعدد ثقافات أعضاء الفريق البرمجي.

تم تصميم هيكلية نظام معالجة البيانات DPS2000 ليكون مرناً وقادراً على التعامل مع أنواع كثيرة من التطبيقات. وكان ذلك تصميماً أولياً من حيث المتطلبات. وقد ساعد التنوع أعضاء الفريق البرمجي في تحقيق تصميم مرن من حيث المهارات والخبرات. في هذا المشروع، أضافت مساهمة فرق البرمجة الموزَّعة في عدة دول جاذبية إلى عروض المنتجات العالمية.

المراجع

Books

Paulish, Daniel J. *Architecture-Centric Software Project Management*. Boston; MA: Addison-Wesley, 2002.

الفصـــــل السادس عشر

نظام المعلومات المالية

حققت شركة الخدمات المالية (FSI) نجاحاً كبيراً في توفير الحلول البرمجية للخدمات المتعلقة بالبنوك. وبمرور الوقت، لوحظ تغيّر تدريجي في هذا النوع من الخدمات. فقد اندمجت بعض البنوك غير المستقلة والرهن العقاري وشركات الاستثمار ومزودو خدمات التأمين جميعاً بشكل تدريجي ليشكلوا شبكة كبيرة ومتكاملة من مزودي الخدمات المالية. يمكن الآن للعملاء أن يتوجهوا إلى أي مؤسسة ضمن شبكة مالية للحصول على خدمة مالية معينة، ويتم عرض خدمات ومنتجات مالية أخرى عليهم بكل سهولة. وما إنْ يفتحوا حساباً في مؤسسة واحدة، حتى يكون هناك حاجة لفتح حسابات منفصلة في مؤسسات أخرى ضمن نفس الشبكة. ويعمل هذا الحساب كمستودع واحد يخدم المحفظة المالية الكلية للعميل، ويصبح من الممكن أن يستلم العميل بياناً مالياً واحداً بدلاً من استلام عدة بيانات يمثل كل منها خدمة مصرفية واحدة.

وجدت شركة الخدمات المالية أن هذا يمثل فرصة تجارية

سانحة لإنتاج حلول برمجية جديدة ونظام معلومات مالي (FS2000) واحد يتيح تنفيذ عمليات دمج وتدفق للمعلومات خلال جميع المشاريع بشكل سلس. هذا، وقد شكلت الشركة فريقاً للبحث والتطوير (R&D) يتكون من خبراء في هذا المجال ومهندسي برمجيات للبدء على الفور بتنفيذ هذا المنتج.

16-1 متطلبات المشروع الجديد

يعمل الخبراء في هذا المجال ومهندسو البرمجيات ضمن فريق يعرف الكثير عن النموذج القديم الخاص بالبنوك، واكتسبوا خبراتهم من النظام المستقل الذي قامت الشركة ببنائه لخدمة البنوك المستقلة وراهني العقارات والشركات الاستثمارية ومزوّدي خدمات التأمين. وقد تطلعوا إلى الفرصة الجديدة على أنها ستمثل عملية إنتاج عمل متكامل يستطيع نقل المعلومات الخاصة بالعميل وجعلها متوفّرة في هذه المشاريع بسهولة. ونظراً إلى اختلاف كل هيئة مالية عن الهيئات الأخرى في طريقة انتقال المعلومات وعرضها والدخول إليها، كان من الضروري إنشاء آلية مرنة قادرة على التكيف من أجل العمليات التجارية ومواصفات واجهة التطبيق.

لم تقم الشركة بإحضار خبراء في هذا المجال ممن يمتلكون خبرة في نموذج العمل التجاري الجديد لاعتقادهم بأنهم يمتلكون معرفة عميقة في مجال المنتج. واعتبر فريق المشروع أن المتطلبات الوظيفية متقاربة لحد بعيد باستثناء القليل من الوظائف الخاصة بالمنتج؛ أي إنه يجب فقط أن تكون الطريقة والمكان الذي يتم فيه استخدام هذه الوظائف مرنَين.

ونظراً إلى أن الفريق الذي يتكون من خبراء ومهندسي البرمجيات كان صغيراً جداً، لم يعتبر أنه من الضروري إنشاء أي

متطلبات رسمية ونموذج لمجال العمل. وتم استخدام أسلوب العصف الذهني للمعلومات كآلية مبدئية للوصول إلى جزئيات جميع العمل المطلوب.

نتج من اتباع أسلوب الترميم وثقافة العمل التي انتهجتها الشركة مجموعة من العواقب. **الأول**: على الرغم من أن رؤية الشركة والفرصة المتاحة في السوق كانتا رائعتين، غير أنه لم يكن كافياً إنجاز نظام مرن ومتكامل فقط. كان هناك الكثير من القطاعات في السوق حيث تتراوح بين المؤسسات الصغيرة والكبيرة، وقد كان من الضروري أن يقيس النظام احتياجاتهم. وقد كانت مكاتب بعض المؤسسات الكبيرة موزَّعة بشكل كبير وتختلف في ما بينها في نظام التوقيت؛ وكان على النظام أن يأخذ بعين الاعتبار هذا الانتشار عندما يتم توزيعه في منطقة العمل. ولم تحتج المؤسسات الصغيرة إلى جميع إمكانيات النظام الجديد؛ واحتاج النظام إلى إعادة تجزئته إلى مجموعة من المكوّنات توفِّر مرونة في عملية توزيع النظام بأسلوب تدريجي وفق الحاجة. ولم يرغب معظم العملاء في تحمل مسؤولية إدارة النظام الخاص بهم؛ إذ كان يجب أن يتم توزيع النظام والتعامل معه بطريقة فعالة من حيث التكلفة من مركز المعلومات الخاص بالشركة. وكان هناك مجموعة من العمليات التجارية التي تعتبر مهمة وحرجة بالنسبة إلى العملاء؛ وكان على النظام أن يكون متوفّراً للاستخدام في أي وقت بشكل مؤكد.

تمثّل العيب **الثاني** والأكثر أهمية بعدم جمع المعلومات المتعلقة والمتطلبات الخاصة بهذا المجال، وقد كان من الصعب عليهم التواصل في ما بينهم. وكان نتيجة ذلك هو الشعور بنقص حاد في المعلومات. وفي النهاية امتلكت شركة متعددة الجنسيات MNC شركة الخدمات المالية.

16-2 تنظيم عملية التطوير

رأت الشركة متعددة الجنسيات أن الاستراتيجية الأولية التي اتبعتها شركة الخدمات المالية تعتبر مكملاً مهماً لمنتجاتها وخدماتها الحالية، ما جعلها قادرة على إيجاد حلول متكاملة لاحتياجات المؤسسات المالية المنتشرة حول العالم. إضافة إلى ذلك، ستتيح لهم شركة الخدمات المالية فرصة امتلاك حصة مهمة من السوق في أميركا الشمالية.

حالما تمت عملية الامتلاك، رغبت الشركة متعددة الجنسيات بتوسيع مجال عمل نظام المعلومات المالية (FS2000)، إذ أرادت أن تجعل منه نظاماً دولياً. وكان انطباع الشركة متعددة الجنسيات بأن تصور شركة الخدمات المالية للمرونة سوف يمنحهم القدرة على ملاءمته ليناسب أي منطقة حول العالم.

كما وفَّرت الشركة متعددة الجنسيات لهذا العمل مجموعة دولية من مهندسي البرمجيات، إذ اعتُقد أن ذلك سوف يجعلهم قادرين على بدء عمل برمجي متوازٍ يقلص الزمن اللازم لطرح المنتج في السوق. وبالإضافة إلى وجود الفرق في المواقع المختلفة داخل الولايات المتحدة في الوقت الحالي، شكّلت الشركة متعددة الجنسيات فريق تطوير برمجيات في مواقع مختلفة، هي النرويج والسويد وألمانيا والمملكة المتحدة والهند. وقد تم تجزيء المتطلبات، كلّ وفق وظيفتها، إلى وحدات متعددة وتوزيعها حول العالم لإجراء العمليات البرمجية. ونفّذ الخبراء في هذا الموضوع الموجودون في الولايات المتحدة عمليات التحليل والتصميم بينما تمت العمليات البرمجية في المواقع التي تعمل عن بُعد في دول أخرى. وبعد انتهاء العمليات البرمجية على الوحدات أُعيدت إلى الولايات المتحدة لإجراء عمليات الدمج.

ومع ذلك، لم يكن هناك مجموعة من المتطلبات الرسمية أو النماذج التي تقيس بشكل منهجي تغير وتوسع الحلول الخاصة بعمل نظام المعلومات المالية (FS2000). ونتيجة لعدم وجود هذين الأمرين، أنشأت فرق العمل لمشروع نظام المعلومات المالية (FS2000) حلاً مرناً بدرجة كبيرة بحيث يجعل من الممكن تطبيق أي مواصفات للعمليات الخاصة بأي عمل تجاري وكذلك عرض المعلومات المتعلقة بهذا العمل على واجهة التطبيق الخاصة بالمستخدم. وقد كان على الفرق أن تنشئ منفذاً للبرمجيات حسب طبيعة هذه المواصفات، إذ كلما أصبحت المواصفات أكثر تعقيداً، ازداد تعقيد منفذ البرمجيات أيضاً. وفي النهاية كان هذا العمل بمجمله مكلفاً من حيث الاختبار والمعالجة والصيانة وعمليات الدعم؛ كما إن أداء المنتج كان ضعيفاً.

وفي الوقت نفسه، استمرت فرق العمل في المشروع بالنمو لتسريع العمليات البرمجية. وقد أدى هذا إلى وجود مسارات طويلة للتواصل بين الموقع المركزي في الولايات المتحدة ومواقع البرمجة التي تعمل عن بُعد. ولذلك، واجه المسؤولون عن المشروع صعوبة كبيرة في تكوين فهم مشترك لهذه الأمور وتنسيق أنشطة العمل بين فرق المشروع.

كانت الشركة متعددة الجنسيات متحمسة للتقدم في العمل، واعتقدت أن بإمكانها البدء في عملية تطوير للمشروع عن طريق جمع المعرفة في مجال نظام المعلومات المالية (FS2000) والمتطلبات الخاصة به. فقد تقوم عملية الحصول على المعرفة بتيسير عمليات التواصل وتساعد في المهام الصعبة المتعلقة بنقل المعلومات ذات العلاقة إلى الفرق التي تعمل عن بُعد. وهنا واجهت الشركة متعددة الجنسيات عقبة أخرى. وتمثلت خيبة أمل الشركة في أنه كان هناك

القليل من الوثائق التي توضح طريقة جمع العمل في مشروع نظام المعلومات المالية (FS2000). كما وُجد أيضاً أن فريق مشروع نظام المعلومات المالية (FS2000) كان في الحقيقة عبارة عن عدد من المجموعات المختلفة الصغيرة يركِّز كل منها عمله في مجال الخبرة لديه مع عدم وجود أي تواصل بينهم في الأغلب. وجعل هذا كل فريق يعتقد أن وظيفة المشروع تستهدف فقط المجال الذي يعمل فيه دون أدنى فكرة عن الآلية التي سوف يتم من خلالها دمج أجزاء العمل لتكون مشروعاً واحداً وهو نظام المعلومات المالية (FS2000). والأسوأ من ذلك، استخدم كل فريق مجموعة من التقنيات المفضلة لديه لإنجاز العمل الخاص به.

16-3 الهيكلية

بينما كانت الشركة متعددة الجنسيات تحاول فهم نظام المعلومات المالية (FS2000) وجعله يعمل، خطرت لنائب رئيس البحث والتطوير فكرة؛ «لماذا لا نقوم بجمع مكوّنات التطبيق الحالي التي يمكن إعادة استخدامها وتكوين النظام تدريجياً بتركيب مكوّن واحد في كل مرة؟» وتم تعيين مسؤول عن الهيكلية وبدأت الجهود تنصبُّ في تحديد أقل مجموعة من المكوّنات التي يجب أن يحتويها النظام بحيث يمكن بيعها في السوق.

طرح مسؤول الهيكلية على فريق عمل إدارة المشروع هذا السؤال، وهو «ما هي أقل مجموعة مكوّنات يمكن الحصول عليها بحيث تكون قابلة للبيع؟»، وقد فوجئ بأن فريق عمل إدارة المشروع لم يشارك نهائياً في مشروع نظام المعلومات المالية (FS2000). وكانت الاستراتيجية الوحيدة لدى هذا الفريق هي بيع النظام الجديد عندما يكون جاهزاً إلى عملائهم المعروفين. ولم يكن هناك أي

اهتمام بالمؤسسات الصغيرة لأن شركة الخدمات المالية، التي تم تملكها، كانت تبيع منتجاتها للمؤسسات الكبيرة فقط. أعتُبر نظام المعلومات المالية (FS2000) نظاماً متكاملاً، ولذلك لم يضع فريق العمل بعين الاعتبار أن يتم التسليم للنظام بشكل تدريجي عن طريق بيع أجزاء منه فقط في كل مرة.

في الوقت الذي أصيبت به الشركة متعددة الجنسيات بخيبة الأمل من الأخبار التي تلقتها من مسؤول الهيكلية، لم تكن الشركة تنوي الاستسلام بعد. فربما ببعض الحظ، يمكن أن يكون هناك أصول مشتركة بين مجموعة حزم الوظائف الكلية لنظام المعلومات المالية (FS2000). وقد يشكل هذا نقطة بداية جيدة، في حال توفّره.

قام مسؤول الهيكلية بدراسة مجموعة البحث والتطوير للوصول إلى فهم الهيكلية. واعتبر كل فريق صغير داخل مشروع نظام المعلومات المالية (FS2000) فريقاً افتراضياً للبحث والتطوير بحيث أوجدوا الحلول للجزء الخاص بهم من العمل باستخدام نماذجهم الخاصة وتقنيتهم المفضلة. وكانت أفضل الوثائق التي حصل عليها مسؤول الهيكلية مجموعة من شرائح العرض (Power Point). وبالتالي، تحولت الجهود من أجل تحديد أصول مشتركة إلى عمليات لتوثيق الهيكلية.

لم تكن الشركة متعددة الجنسيات مستعدة للاستسلام بعد، فطلبت الشركة من مسؤول الهيكلية لديها إكمال عملية توثيق الهيكلية للوصول إلى نتيجة في عملية توحيد الجهود المشتتة لعمل مجموعة البحث والتطوير للوصول إلى نظام معلومات مالية (FS2000) متكامل. وبعد عدة شهور من العمل الشاق، ظهرت أخبار أخرى مخيبة للآمال، فقد جمع كل فريق من مجالات التخصص الأجزاء الخاصة بحلول هذا المجال الخاص به معاً في نموذج بيانات خاص

مستقل، أي إنه تم تمثيل الأمور والمفاهيم المتشابهة بطرق مختلفة في كل نموذج من نماذج البيانات. ولم يكن هناك فهمٌ مشتركِ أو توافقُ لأي من المفاهيم، حتى بالنسبة إلى الأمور الأساسية كحساب العميل. وأدركت الشركة متعددة الجنسيات الآن أن لديها مشكلة مُمِلَّة.

16-4 إعادة هيكلة المؤسسة

لاحظت الشركة متعددة الجنسيات أن التزام أعضاء فريق عمل مشروع نظام المعلومات المالية (FS2000) كان ضعيفاً في ما يتعلق بتعاونهم مع بعضهم بعضاً، ما أدى إلى أن يصبح مشروع نظام المعلومات المالية (FS2000) نظاماً وهمياً، إذ إن أعضاءه ينتمون إلى أصول تنظيمية مختلفة، ويمثلون مجال العمل الخاص بهم. فقد كانوا جميعاً يخدمون مصالح التنظيم الذي ينتمون إليه، لكن بوجود حافز ضعيف جداً للعمل من أجل رؤية مشتركة. ونظراً إلى أنه لم يكن هناك أي توضيح للرؤية، كوّن الجميع فهماً جزئياً وغير متماسك عن فكرة مشروع نظام المعلومات المالية. وكان هناك حاجة لهيكل تنظيمي جديد.

في البداية، قامت الشركة متعددة الجنسيات بالتأكد من أن فريق عمل إدارة المشروع قد قام بجهود مناسبة وكافية في عملية تحليل السوق. فقد قام الفريق بتحديد المنتجات المناسبة لشرائح مختلفة من السوق وبأقل مجموعة من مكوّنات نظام المعلومات المالية (FS2000) التي يمكن بيعها. وقام أيضاً بتحديد التطويرات المتزايدة التي ستُجرى على أقل مجموعة من المكوّنات للوصول إلى حل متكامل لنظام المعلومات المالية (FS2000).

أنشأت الشركة كذلك مجموعة استراتيجية ومستقلة لعمليات

التصميم الهيكلي تتكون من الخبراء المهمين في هذا المجال ومهندسي برمجيات يترأسهم مدير عمليات البحث والتطوير. وتم تكليف هذه المجموعة بالحصول على تصميم هيكلي عام وموثّق بشكل جيد لجميع الأجزاء التي تتكون منها مجموعة المنتج لنظام المعلومات المالية (FS2000). وكان هدف الهيكلية هذه تحديد الأصول المشتركة لجميع أجزاء مجموعة المنتج، وكانت وظيفة المجموعة إدارة عمليات البرمجة التي يقومون بها.

تم إنشاء مجموعات منفصلة لتعمل كل منها في مجموعة فردية من مجموعات نظام المعلومات المالية (FS2000). ويترأس كل مجموعة من هذه المجموعات مدير للمنتج يقوم بالإشراف على عمليات التحليل والتصميم للجزء الخاص من مجموعة وظائف المنتج. وتم تقسيم الوظائف إلى مجموعة من المكوّنات الصغيرة يمكن التحكم بها، ومحددة بشكل جيد بحيث يمكن برمجتها في نفس الوقت بواسطة فرق من مهندسي البرمجيات التابعين لمنظمات الشركة متعددة الجنسيات الموزَّعة حول العالم.

كانت عملية دمج أجزاء المنتج لإيجاد حلول للشرائح المختلفة من السوق مسؤولية المجموعة الاستراتيجية لعمليات التصميم الهيكلي.

16-5 إنجاز التكامل

لم يكن فريق مشروع نظام المعلومات المالية (FS2000) موزعاً وفق مجالات التخصص فقط، بل كان نموذج البرمجة باستخدام الكائن (object-oriented) غير مألوف لديه أيضاً. وأدى ذلك للوقوع في العديد من المشاكل. فقام الفريق باتباع فكرة تحليل الكائن البرمجي بشكل جيد لكن لم يستطع تجميع الطبقات المترابطة إلى

مكوّنات ذات وجهات محددة بشكل جيد، لذلك قام بإخفاء هذه الطبقات عن العالم الخارجي. وأدى ذلك إلى عمليات اقتران وترابط موسعة بين معظم الطبقات، منتجاً حلاً برمجياً جامداً غير مرن وهش أيضاً. ونتج من عدم وجود وحدات برمجية صعوبة في إنشاء مجموعات من وظائف المنتج يمكن بيعها في السوق لتعمل كمنتج يؤدي وظائف كاملة.

نتيجة تجزيء مشروع نظام المعلومات المالية (FS2000) وفقاً لمجالات التخصص وطبيعتها المتماسكة، ظهرت تحديات للمجموعة الاستراتيجية لعمليات التصميم الهيكلي المكوّنة حديثاً. وركّز كل مجال من مجالات التخصص على مجموعة من الوحدات كانت كبيرة ليتم التعامل معها في مشروع واحد. ويجب أن يتم التأكد من أن الطريق الذي اتبعته المجموعة لإنهاء العمل بهذه التجزئة وإنشاء وحدات للحلول يمكن التحكم بها بشكل سهل، ليس فقط ليلاقي دعماً من جميع الأطراف، بل لجعل إنشاء رؤية مشتركة للبنية الهيكلية أمراً ممكناً أيضاً.

قررت المجموعة استخدام النسخة المعيارية من برمجية (J2EE). وتم اعتماد هذا الاختيار لعدة أسباب: **أولاً**، اعتقدت المجموعة أن استخدام مجموعة معيارية سترحب به التنظيمات المختلفة التي تعمل على المنتج والتي كانت تشارك سابقاً في النقاشات التي تدور حول اختيار الهيكلية الأفضل. وتم قضاء قدر كبير من الوقت في عرض الأسباب التي جعلتهم يقومون بهذا الاختيار.

ثانياً، رأت المجموعة أنه من الأفضل استخدام الطاقات الموجودة أصلاً في الشركة للعمل على الأمور الأساسية، والتي تتمثل في إيجاد الحلول للمجال المالي وليس إيجاد نظام تقني

للعمليات البرمجية على المنتج لأنه من السهل الحصول على مثل هذه الأنظمة في السوق. وسوف يجعل ذلك الشركة تركز على الإبداع وتطوير منتجات جديدة وطرحها في السوق خلال وقت قصير.

ثالثاً، سيكون من المفيد في مشاريع التطوير الموزَّعة بشكل واسع، والتي يتم فيها تكوين مجموعات برمجية من مختلف دول العالم، أن يكون الاختيار معيارياً لأنه سوف يضمن وجود فهم مشتركٍ للبنية الهيكلية بشكل أسرع. على الأقل بالنسبة إلى جوانبَ برمجية (J2EE) تكون معيارية للجميع بحيث يمكن الحد بشكل ملحوظ من المقابلات المباشرة، وبالتالي، من كلفة السفر حول العالم.

أخيراً، يمكن للفرق التي تعمل على المنتج أن تستخدم نماذج برمجية (J2EE) في عمليات تصميم المنتج وبرمجته. وقد يكون هناك بالطبع حاجة لبعض العمليات الإضافية لضمان حلول للوحدات تكون القدرة على التحكم بها أكبر.

16-6 دروس مستفادة

كان أكثر الأمور التي واجهت الشركة متعددة الجنسيات أهمية، عندما امتلكت شركة الخدمات المالية، عدم وجود عمليات توثيق جيدة للمتطلبات ونموذج مجال العمل. ونتيجة لعدم إنشاء هذه المهام ظهرت مشكلتان رئيستان: **الأولى**، أغفلت شركة الخدمات المالية كلياً نقطتين تتعلقان بالجودة حين وضع النماذج الخاصة بأجزاء المنتج، وهما القدرة على التغيير والقابلية على التوسع. وعلى الرغم من أن الحلول المتعلقة بنظام المعلومات المالية (FS2000) مرنة وقادرة على التكيف، غير أن مرونتها وقدرتها على التكيف كانت

محدودة في أسواق أميركا الشمالية. وقد واجهت الشركة متعددة الجنسيات وقتاً عصيباً في تجربة الطريقة التي يمكن من خلالها توسيع مجال عمل هذا النظام ليصبح مناسباً للعمل في الأسواق العالمية. أما **المشكلة الثانية** والأكثر أهمية من ناحية التطوير الموزع، فلم يكن من الواضح كيف يمكن إتمام المهمة الصعبة المتعلقة بنقل المعرفة إلى المواقع التي تعمل عن بُعد مع عدم وجود هذه المهام.

إضافة إلى ما تقدم، لم يكن لنظام المعلومات المالية (FS2000) توثيق جيد لهيكليته. وفي الحقيقة، لم يكن هناك هيكلية واحدة، بل مجموعة من البنى الهيكلية المنفصلة لكل منتج من مجموعة منتجات نظام المعلومات المالية (FS2000). كما افتقر النظام إلى تصميم للوحدات، ما جعل من الصعب فصل الأصول الأساسية العامة الموجودة في جميع أجزاء المنتج. وأدى ذلك أيضاً إلى صعوبة عملية إنشاء مجموعة من المكوّنات ينتج عنها منتجٌ يعمل بشكل كامل ومناسب لجميع المؤسسات المالية من الصغيرة إلى الكبيرة. ونتج من ذلك أيضاً عدم قدرة الشركة متعددة الجنسيات على امتلاك استراتيجية يمكن بواسطتها إنشاء الحلول بطريقة تدريجية ومتزايدة عن طريق إضافة المكوّنات بشكل مستمر على شكل مراحل. أما أكثر التأثيرات أهمية في عمليات التطوير الموزَّعة فكان عدم وجودة توثيق للهيكلية مما صعَّب عملية النشر وإنشاء رؤية مشتركة للبنية الهيكلية بين الفرق التي تعمل عن بُعد؛ ولم تجعل الطبيعة الصلبة من الصعب توزيع العمليات البرمجية لنظام المعلومات المالية (FS2000) وحسب، بل وجعل حجم الوحدات المتعلقة بمجالات التخصص المختلفة وتوابعها من الصعب التعامل مع تطوير العمليات البرمجية.

لم تكن الهيكلية التي اتبعتها الشركة المتعلقة بجهود البحث والتطوير لنظام المعلومات المالية (FS2000) كافية أيضاً. وكان فريق

البحث والتطوير افتراضياً ولم يكن لديه حافز لتحقيق التصور للحلول الجديدة. وكان أعضاء الفرق المختلفة منحازين لصالح منظمتهم الأم، وأدى هذا إلى تكوّن فرق صغيرة تعمل في المشروع منغلقة حول العمل في مجال تخصصهم نتج عنه ما كان يجب أن يتجنبه نظام المعلومات المالية (FS2000)، وهو إنشاء أنظمة مستقلة غير مترابطة. وأصبح الأمر أكثر سوءاً كلما زاد عدد أعضاء الفريق الذي يعمل في مجال تخصصه. وأصبحت مسارات التواصل بين الفرق طويلة جداً، ما صعّب من عمليات تنسيق الأنشطة بين الفرق وإيجاد فهم مشترك في ما بينهم.

16-7 الملخص

في حالة نظام المعلومات المالية (FS2000)، أدى الحجم الكبير لكل مكوّن من المكوّنات والحجم الكبير لعدد أعضاء الفرق التي تعمل في أحد مجالات التخصص والحجم الهائل للمشروع بأكمله إلى ظهور تعقيدات في عمليات التطوير الموزَّعة. إضافة إلى ذلك، أدى عدم وجود عمليات توثيق المتطلبات والهيكلية إلى وجود صعوبة في تكوين تصور عام ونقل هذا التصور إلى مهندسي البرمجيات الجدد الذين تم ضمهم إلى المشروع بعد امتلاك شركة الخدمات المالية. عززت هذه الدراسة للحالة فكرتنا بأن المشاريع الصغيرة تكون عادة أفضل. وعلى كل حال، تتطلب الاحتياجات الجديدة للشركة إلى منتج نظام المعلومات المالية (FS2000) ضخم ومتعدد المتغيرات، ما أدى إلى ظهور تعقيدات عديدة. ويقوم الفريق حالياً بإعادة تحديد الهيكلية، ويقوم باختبار طرق لإعادة تشكيل الصيغة البرمجية بحيث يتم تجزيئها إلى مكوّنات أصغر يمكن دمجها في نظام العمل الجديد.



الفصـــل السابع عشر

نظام إدارة المباني الآلي

نظام إدارة المباني الآلي (BAS) هو نظام لإدارة أجهزة المراقبة والتحكم بها ضمن أحد المباني أو في محيط مجموعة من المباني. ويستخدم نظام إدارة المباني الآلي في إدارة مختلف تطبيقات العمل التلقائية ضمن المبنى كالتدفئة والتهوئة والتكييف والأمن والحريق والتحكم في الدخول والتطفل.

17-1 تمهيد

تم إنشاء الوثائق الخاصة بمشروع نظام إدارة المباني الآلي ومشاريع إدارة المحطات التي سبقته في دراسة الحالة لشركة سيمنز بواسطة هيرسليب (Herbsleb, 2005). وقد تمت عمليات البرمجة في سبعة مواقع تابعة لشركة سيمنز في سبع دول مختلفة هي سويسرا وألمانيا وإيطاليا وأستراليا وسلوفاكيا والهند والولايات المتحدة.

تصف هذه الدراسة الممارسات التي اتُبعت خلال مرحلتي البدء والتطوير لمشروع نظام إدارة المباني الآلي. وخلال مرحلة التحضير،

تم تشكيل فريق صغير لتصميم الهيكلية يتكون من مصممين من مواقع مختلفة. وقد عمل الفريق في الموقع المركزي الموجود في الولايات المتحدة وفقاً لبرنامج زمني محدد، يتضمن العمل لثلاثة أسابيع في الموقع المركزي، ومن ثم أسبوعين في مواقعهم الأصلية. وقد تم نقل المصمم الموجود في أستراليا إلى الولايات المتحدة الأميركية حتى انتهاء فترة مرحلة التحضير. وقد قام فريق المصممين بإنجاز عمليات التحليل الشامل وتحليل المتطلبات ذات الأهمية الكبيرة؛ كما أنشأوا شجرة المفاهيم وقاموا بمراجعة الهيكلية في نهاية المرحلة. وإضافة إلى ذلك، قام مدير المشروع بوضع تقديرات للجهد الذي تحتاجه العمليات البرمجية للمشروع والوقت الزمني اللازم لإتمامه وفقاً للخريطة المفاهيمية التي تم إعدادها.

خلال مرحلة التطوير، تم زيادة حجم الفريق لإنجاز متطلبات المنتج وهندسة المتطلبات وتصميم الهيكلية وإدارة البرنامج وتصميم واجهة التطبيق وتحديد العملية وعملية إنشاء النماذج. وخلال هذه المرحلة، تم استخدام أسلوب «الفريق الافتراضي» بشكل متزايد، حيث يعمل أعضاء الفريق بشكل رئيس في مكاتبهم الأصلية، ويقومون بحضور الاجتماعات الدورية في المواقع المختلفة التي تم تحديد مواعيدها مسبقاً. على سبيل المثال، يجتمع فريق إدارة البرنامج شهرياً، وهو في حالة تنقل دائمة بين المواقع في سويسرا والولايات المتحدة الأميركية. وقد تم نقل فريق العمل المكوّن من المواقع الخارجية والمواقع التي من المتوقع أن تتم فيها معظم العمليات البرمجية خلال مرحلة التحضير إلى موقع في الولايات المتحدة الأميركية حيث باشروا في أنشطة إنشاء النماذج، وتم تدريبهم على الهيكلية وعلى مجال العمل التجاري عن طريق فريق التصميم.

17-2 التحليل الشامل

تم إنجاز التحليل الشامل في مشروع نظام إدارة المباني الآلي كمهمة من مهام فريق التصميم. وكان أحد العوامل التنظيمية المهمة والمؤثِّرة التي برزت خلال عمليات التحليل الشامل أن هناك حاجة إلى فريق عمل برمجي لتنفيذ نظام إدارة المباني الآلي في الوقت المحدد أكبر من فريق العمل الموجود حالياً. وعلى الرغم من وجود العديد من المواقع المهتمة بالمشاركة في العمليات البرمجية لنظام أتمتة المباني؛ غير أنه، مع ذلك، وحتى لو اجتمع جميع مهندسي البرمجيات من كافة المواقع، فإنهم لن يكونوا كافين لإنجاز هذا العمل. ومردّ هذا النقص في فريق العمل كان بشكل أساسي نتيجة صعوبة تحرير فريق العمل الذي يعمل في الوقت الحالي في مشاريع مُدِرَّة للدخل باستخدام منتجات سابقة. إضافة إلى ذلك، كان هناك عامل مؤثر في سرعة طرح المنتج في الأسواق نتج منه جعل حجم فريق البرمجة أكبر من حجمه مقارنة بما كان عليه في مشاريع برمجة سابقة. لذلك، تم تحديد استراتيجية المشروع مبكراً بأنه سيكون من الضروري الحصول على مصادر برمجية من الخارج.

يتطلب نظام إدارة المباني الآلي دمج التطبيقات ذات المجالات المختلفة لتعمل كمحطة عمل واحدة. ونظراً إلى أن هذه المجالات قد نفّذت كوحدة تخصصية مركزية بين المواقع البرمجية في الشركة، فقد نتج من ذلك تعديل استراتيجية المشروع للاستفادة من الخبرات في هذا المجال الموجودة في مواقع الشركة الموزَّعة. لذلك، أصبح من المعروف مقدماً أنه سيكون هناك حاجة إلى فريق عمل من المواقع البرمجية المختلفة التابعة لشركة سيمنز لتنفيذ نظام إدارة المباني الآلي.

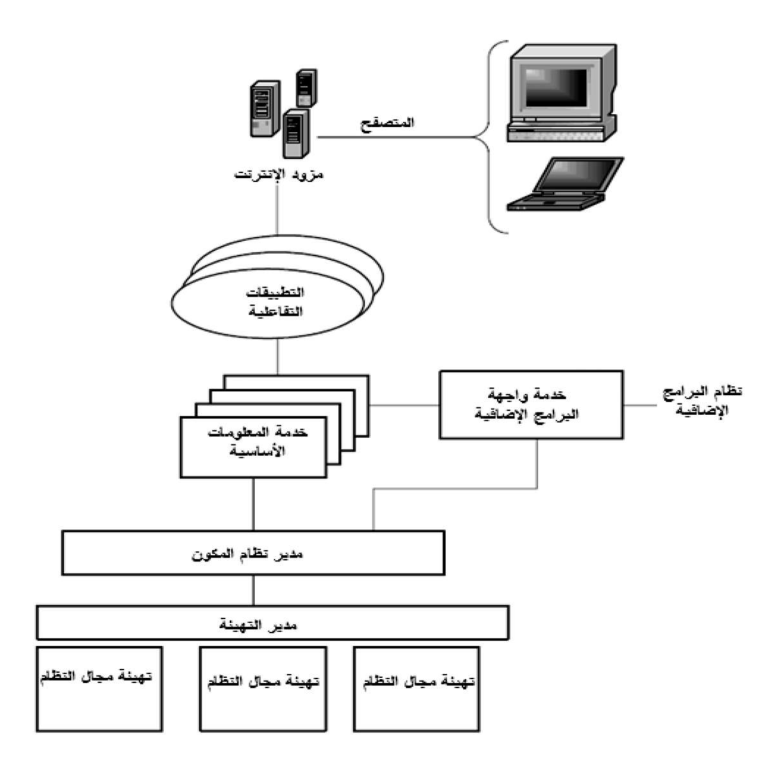
لقد تم وصف الكثير من الممارسات المتبعة في نظام إدارة

المباني الآلي في هذا الكتاب، خصوصاً استخدام مدير التزويد للمساعدة في التعامل مع المصادر الموجودة في مواقع البرمجة التي تعمل عن بُعد. هذا، وبما إن العمل في المشروع ما زال مستمراً، فلن يكون معروفاً قبل سنوات إذا كان سينتج من الممارسات المتبعة مشروع ناجح ومنتج جيد. وعلى كل حال، اعتُبر العديد من الممارسات المتبعة خلال مرحلتي البدء والتطوير من أفضل الممارسات على أساس أنه تم اتباعها في مشاريع سابقة.

17-3 هيكلية نظام إدارة المباني الآلي

تم تصميم هيكلية نظام إدارة المباني الآلي (BAS) بحيث يمكن إجراء العمليات البرمجية على المكوّنات عن طريق فرق موزَّعة بما يتناسب مع نطاق الهيكلية. ويمكن إنجاز التطبيقات الخاصة بالمنتجات السابقة وعملية نقلها عن طريق المراكز المتخصصة. وسوف تتطور المواقع الخارجية لتصبح مواقع متخصصة بمرور الوقت؛ ولكن في مراحل عمل البرمجة الأولية، ستقوم الفرق بتطوير البنية التحتية مثل المكوّنات الخاصة بإنشاء قاعدة العمل. ولذلك، سوف تكون الهيكلية الناتجة عملية ومنشأة وفق المقاييس لأنه كان هناك مشاركة دولية في عمليات البرمجة منذ البداية. إضافة إلى ذلك، طُلب من المصممين أن تكون المكوّنات صغيرة. وطُلب من المصممين، بالتحديد، أن يكون حجم المكوّن الواحد أقل من 60 ألف سطر من الشيفرة البرمجية (KLOCs) بلغة برمجة #C. وكان هذا ناتجاً من الرغبة في أن يكون حجم الفريق البرمجي الذي يعمل على هذه المكوّنات عن بُعد صغيراً.

يوضح الشكل 17-1 وصفاً للبنية الهيكلية لنظام إدارة المباني الآلي.



الشكل 17-1: الهيكلية العليا لنظام إدارة المباني الآلي (BAS)

تم تنفيذ خطة المشروع باتباع الممارسات التقديرية التي تم وصفها في الفصل الثامن. فقد كان مدير المشروع يعمل جنباً إلى جنب مع رئيس المصممين ليتم تقدير حجم الشيفرة البرمجية المصدرية لكل مكوّن من المكوّنات التي كان من المتوقع تطبيقها في المشروع خلال مرحلة التحضير. وقد تم تجزيء العمليات البرمجية المتوقعة للمشروع إلى وحدات برمجية، وتم تحديد الجهد المتوقع بذله وحجم الفريق اللازم لذلك لكل مكوّن من المكوّنات. وقد تم تخصيص مجموعة من المكوّنات للوحدات البرمجية وللمواقع البرمجية المتوقعة أيضاً. وتم تحديد الموارد الداخلية اللازمة لحوالى

نصف المهام البرمجية التي سيتم تنفيذها. وتم افتراض أن النصف الثاني ستقوم المواقع الخارجية بتوفيره، وتم بذل جهد كبير في التخطيط للجهود التي سيتم بذلها في تأهيل هذه المواقع الخارجية وإحضار الأعضاء المهمين إلى الفريق المركزي خلال المراحل المبكرة بحيث يصبحون قياديين لفرق برمجة المكوّنات التي تعمل عن بُعد.

لقد بُذل جهد لا بأس به في اختبار إمكانية إعادة استخدام المكوّنات بدلاً من إعادة إنشاء بعض المكوّنات الضرورية. وطرحت هذه التحديات العديد من الأسئلة التي تتعلق بمناسبة الوظائف والتقنيات؛ ولكن بسبب النقص الكبير في الموارد المتاحة، تم بذل كل جهد متاح لإيجاد مكوّنات مناسبة لإعادة استخدامها أو تعديلها بشكل طفيف بحيث يمكن إعادة استخدامها بدلاً من إعادة عملية البناء من الصفر.

كذلك، وبسبب النقص الكبير في الموارد البرمجية المتاحة، تم طلب العديد من المصادر من الخارج. وعلى الرغم من أنه من المعروف أنه سينتج من إضافة مواقع برمجية جديدة تعقيدات في إدارة المشروع، وُجد أن استخدام شركتين خارجيتين سوف يساعد في تقليص المخاطر السياسية وتوفير مرونة أكبر عندما تكون هناك حاجة لتوظيف عدد كبير من المبرمجين.

بذلت جهود كبيرة مقدماً في عمليات البرمجة في كل من المواقع الخارجية وفي موقع داخلي ثالث. وساعدت هذه الجهود الكبيرة في تتبع واختبار العمليات البرمجية الشاملة ليتم استخدامها في مرحلة التحضير. وتم تحديد أهداف كل مرحلة شهرية خلال فترة الأربعة شهور للمرحلة التجريبية.

17-4 التخطيط للمشروع

تم التخطيط للمشروع باستخدام ممارسات التقويم التي تم شرحها في الفصل الثامن. وعمل مدير المشروع جنباً إلى جنب مع رئيس التصميم لوضع تقويم لحجم شيفرة البرمجة اللازمة لكل عنصر يُتوقع تطبيقه للمنتج خلال مرحلة البناء. وتم تقسيم مشروع التطوير المتوقع إلى فترات برمجية تكرارية وتم تحديد تقويم الجهد ومعدل حجم فرق العمل لكل من مكوّنات المنتج. وتم توزيع المكوّنات على الفترات البرمجية التكرارية وعلى مواقع التطوير المتوقعة. كذلك تم تحديد الموارد المحلية اللازمة لتنفيذ حوالى نصف المهام. وأما النصف الآخر فقد كان متوقعاً أن يتم تنفيذه في المواقع البعيدة، وقد تم بذل الجهد في وضع خطط جديرة بالاعتبار لتأهيل هذه المواقع البعيدة وإحضار الموظفين الأساسيين للعمل ضمن الفريق المركزي خلال المراحل الأولى من المشروع بحيث يصبح أعضاؤه قادة لفرق التطوير التي تعمل عن بُعد. وبُذلت جهود مهمة في اكتشاف إعادة استخدام المرشحين لاكتساب بعض العناصر اللازمة بدلاً من بنائها. قادت هذه التحقيقات إلى إثارة العديد من التساؤلات في ما يخص الأمور الوظيفية والتقنيات الملائمة؛ لكن وبسبب العجز الكبير في الموارد المتاحة، تُبذل جهود هائلة في العثور على العناصر الملائمة التي يمكن إعادة استخدامها أو تعديلها لتصبح ملائمة لإعادة الاستخدام بدلاً من بنائها من الأساس. إضافة إلى ذلك، ونظراً إلى العجز الكبير في الموارد المتاحة للتطوير، يتم الاستعانة بالمصادر الخارجية وهو أمر مزعج. إن إضافة مواقع تطوير إضافية يضفي المزيد من التعقيد إلى إدارة المشروع، ومع ذلك أُخذ هذا الأمر في الاعتبار، فكان الشعور السائد أن الاستعانة بشركتين خارجيتين سيساعد في الحد من المخاطر الناجمة عن الأمور السياسية ويوفر

مرونة إضافية عندما يكون عدد المبرمجين المطلوب توفّرهم كبيراً.

ونظراً إلى وجود موقعين خارجيين وموقع داخلي ثالث، تم إطلاق الجهود التجريبية قبل عمليات التطوير الشاملة. وقد ساعد ذلك في تتبع عمليات التطوير موزَّعة المراكز أثناء مرحلة البناء. وقد تم تحديد أهداف الفترات البرمجية لكل شهر أثناء فترة الإطلاق التي تبلغ مدتها أربعة أشهر.

17-5 إدارة المشروع

يوجد في كل موقع من المواقع المشاركة في عملية تطوير نظام إدارة المباني الآلي مدير للمصادر يعمل محلياً للقيام بالعمليات الإدارية لأعضاء الفريق في الموقع. ويوجد أيضاً مدير عام لبرنامج نظام إدارة المباني الآلي يقوم بقيادة الفريق الإداري للبرنامج. ويوجد لدى المدير المركزي للتطوير مديرا تزويد يعملان لديه في إدارة الروابط في المواقع الخارجية التي تعمل عن بُعد.

وقد كان مسؤول التصميم مسؤولاً عن اتخاذ القرارات التقنية وإيجاد حلول التداخلات التقنية. وهذه الوظيفة مهمة في المشروع إذ تضطلع بمسؤوليات كبيرة. كما يوجد مسؤول عن مهندسي المتطلبات المركزيين توكل إليه مسؤولية إنشاء وتحليل النماذج الخاصة بالمتطلبات التي تم استقاؤها من فريق التسويق الذي قام بوضع طلبات أصحاب المصلحة في التطبيقات ذات التخصصات المختلفة.

وتم تخصيص كل مكوّن من مكوّنات نظام إدارة المباني الآلي إلى فريق برمجي يعمل عن بُعد. ويوجد في الفريق الذي يعمل عن بُعد مجموعة متخصصة تتكون من مهندسي متطلبات مركزيين، ومصممين مركزيين، ومديرٍ للتزويد لذلك الموقع. ويقوم هذا الفريق المتخصص بتحديد أهداف المرحلة الزمنية لفريق برمجة المكوّنات

الذي يعمل عن بُعد، ولكن الفريق الذي يعمل عن بُعد هو الذي يقوم بالتخطيط للوصول إلى هذه الأهداف.

تم إجراء تتبع وضع العمل في المشروع خلال الاجتماعات الشهرية الخاصة بمراجعة العمل التي عُقدت بالتناوب بين موقعي الولايات المتحدة وسويسرا. وتم تحديد برنامج زمني للاجتماعات التي تتم فيها عمليات المراجعة لعدة أيام. ويتضمن البرنامج اجتماعات حول الوضع الحالي للعمل واجتماعات لطرح المعلومات مع وجود أنشطة اجتماعية لتحسين المهام الإدارية وبناء الفريق.

17-6 دروس مستفادة

لقد كان لكثير من الدروس والعبر المستفادة من مشروع نظام إدارة المباني الآلي علاقة بعمليات التواصل بين أعضاء الفرق الموزَّعة. وفي مرحلة مبكرة، قضى فريق التصميم ثلاثة أسابيع في الموقع المركزي، ومن ثم أسبوعين في مكاتبهم الأصلية. وكان لذلك أثر جيد نسبياً، وكان الفريق قادراً خلال هذه الفترة على العمل على تنظيم المهام ذات الطبيعة التعاونية (مثل اجتماعات التصميم)، وأيضاً المهام التي يتم إنجازها بشكل فردي (مثل كتابة وصف الهيكلية). ولدى كل عضو من أعضاء فريق التصميم الصغير هذا مكان مخصص للعمل في الموقع المركزي. وتم تخصيص غرفة اجتماعات للاجتماعات الخاصة بالتصميم. ولذلك، كانت أقصى مسافة بين أعضاء الفريق في الموقع المركزي خلال الأسابيع الثلاثة الأولى، خمسة عشر متراً على الأرجح. ونتيجة لذلك، كان هناك مستوى عالٍ من التواصل بين أعضاء الفريق. إضافة إلى ذلك، تحقق أكبر وأفضل قدر من التواصل عندما كان يجتمع أعضاء الفريق في الموقع المركزي بشكل مستمر لتناول طعام الغداء، وذلك عندما كان الفريق

المحلي يستضيف فريق عمل من المواقع البعيدة في أحد المطاعم في المنطقة. قام مسؤول التصميم الهيكلي بقيادة فريق التصميم وقام بتسهيل مهام العمل وتحديد القرارات التي يتم اتخاذها حول أي أمور متعلقة بالتصميم لم يبت بخصوصها بعد. وقد لاحظنا أن العمل المتعلق بتصميم الهيكلية لا ينتهي أبداً، وأنه من الضروري تجزيئه إلى فترات عمل زمنية لاتخاذ قرارات فعالة حول التصميم (Paulish, 2002). ويعلم المسؤول الفعال عن المصممين الوقت المناسب لاتخاذ قرار حاسم حول التصميم ومتى يستمر بأخذ الآراء حول التصميم لتوفير بدائل مناسبة.

عمل مدير المشروع مع فريق التصميم في تكوين فكرة تقديرية للجهود التي سيتم بذلها في عمليات التصميم والوقت الزمني اللازم لذلك. وكان من الواضح منذ البداية أن مدى المنتج المطلوب لمشروع نظام إدارة المباني الآلي أكبر من المشروع الذي أخذت الشركة على عاتقها أن تنفذه في البداية. استخدم مدير المشروع متطلبات التسويق، ووصف الهيكلية، وأدوات تقدير التكلفة، والبرنامج الزمني، والمعلومات المتعلقة بالجهود التي تم بذلها في مشاريع سابقة من أجل وضع تصور مبدئي للتقديرات المتعلقة بتكلفة التطوير والفترة اللازمة له.

في وقت لاحق، وعندما كبر حجم الفريق ليشمل بعض الأنشطة مثل هندسة المتطلبات والتخطيط لعمليات الاختبار وتحديد العمليات والتخطيط لعمليات الدمج والنماذج الأولية وتصميم واجهة تطبيق المستخدم، تم إعداد الفريق ليعمل كوحدات وظيفية تتكون من فرق عمل في مختلف مواقع البرمجة. لذلك، أصبح الفريق موزعاً في مناطق جغرافية مختلفة، ونشأ بين أعضائه كمّ نسبي أكبر من التواصل وتنسيق أكبر ومشاكل فعلية أكثر مقارنة بفترة الأسابيع الثلاثة

الأولى. وخلال فترة الأسبوعين، عُقدت اجتماعات دورية بين الفرق في أحد مواقع البرمجة في كل مرة. ولوحظ أن الإنتاجية العامة للفريق لم تكن جيدة كما لو كان الأعضاء يبعدون عن بعضهم البعض خمسة عشر متراً على الأكثر خلال فترة الأسابيع الثلاثة الأولى في الموقع المركزي. إن تغيير موقع الاجتماعات بشكل دوري أصاب بعض الأعضاء بالإرهاق جراء السفر، كما عانوا مشاكل جسدية أثرت في مدى يقظتهم وصبرهم، وغير ذلك. ونظراً إلى أن اجتماعات الفرق استمرت لعدة أيام، تم وضع برنامج زمني لزيارات أعضاء الفرق بحيث تكون مدة الرحلة أسبوعاً واحداً على الأرجح. ووفقاً للبرنامج الزمني، على الأرجح، فإن هذا قد يعني السفر خلال العطلة الأسبوعية، ما يؤثِّر في وقت أعضاء الفريق المخصص للعائلة من أزواج أو أبناء. ولتفادي ضياع الوقت المخصص للعائلة، قد يلجأ بعض أعضاء الفريق إلى استقلال رحلات طيران عابرة للقارات بحيث يلتحقون باجتماعاتهم مباشرة. وفي بعض الحالات، قد ينتج من قلة النوم أن تصبح هذه الاجتماعات مرهقة جسدياً.

بمرور الوقت، تم إدراك أهمية تنظيم أعضاء الفريق لتحسين التواصل بين أعضاء الفريق الذين يعملون في الأنشطة التي تحتاج إلى تعاون في المراحل المبكرة للمشروع. لكن، عملية نقل أعضاء الفريق إلى الموقع المركزي كانت معقدة بسبب القيود المتعلقة بالتكلفة المادية والقيود الشخصية للأعضاء المميزين في الفريق. الأمر الآخر الذي نتج من التوزيع الكبير للفرق هو أنه حين وجود أعضاء الفرق في مكاتبهم الأصلية، يتم عادة التواصل بهم للمساعدة في إيجاد حلول لمنتجات سابقة. لذلك، فبالإضافة إلى أن أعضاء الفرق بعيدون عن الموقع المركزي، فقد كانوا أيضاً عادة ما يصبحون يعملون بشكل جزئي في مشروع نظام إدارة المباني الآلي. بعد فترة

من الوقت، تم اتخاذ قرار بأن يتم جمع أعضاء الفرق المهمين ليكونوا موجودين في الموقع المركزي، ولكن نتيجة لذلك اختل البرنامج الزمني للمشروع بسبب الوقت الذي تم فيه اختيار المرشحين وبسبب النقاشات الفردية التي تمت بخصوص عملية نقل هؤلاء المرشحين إلى دول أجنبية. وفقاً لهذه الدراسة ودراسات أخرى أيضاً، نعتقد أن المهام ذات الطابع التعاوني والمتعلقة بهندسة البرمجيات في المراحل الأولى كتصميم الهيكلية، تتطلب فريقاً صغيراً منظماً مشابهاً للفريق الذي كان يعمل خلال الأسابيع الثلاثة الأولى في الموقع المركزي.

بسبب وجود شركتين برمجيتين تابعتين لشركة سيمنز في سلوفاكيا والهند تقومان بتوفير المصادر الخارجية، كان من السهل ملاحظة الفرق في الممارسات التي يقوم بها هذان المزودان. وقد كان هناك حاجة لأن يعمل الأعضاء المهمُّون من المواقع التي تعمل عن بُعد في الموقع المركزي خلال المراحل الأولى بحيث يتم تدريبهم لقيادة فرق برمجة المكوّنات حين عودتهم إلى مواقعهم الأصلية خلال مرحلة التحضير. وقد لمسنا الفائدة الكبيرة من هذه العملية التنظيمية ولم نْستَأْ من التكلفة الإضافية التي تم تحملها نتيجة نقل أحد الأعضاء من دولة ذات تكلفة معيشية منخفضة إلى الولايات المتحدة. وكان الفرق بين الشركتين المزودتين في حجم الدعم الذي يقدمونه لأعضائهم الذين يعملون في دول أجنبية. وقام أحد المزودين بنقل أحد المهندسين الذين يعملون لديه بالإضافة إلى عائلته إلى الولايات المتحدة للتدريب. وقام المزود الآخر بنقل المهندس فقط مع بقاء عائلته في موطنها الأصلي. ولم يكن ذلك تصرفاً جيداً من وجهة نظر أعضاء الفريق المركزي لأن المهندس كان يعود بشكل مستمر إلى وطنه في العطل لزيارة العائلة. وفي أسوأ الحالات، فإن

هذا يؤثّر سلباً في الأهداف المرجوة من وجوده للأسابيع الثلاثة الأولى المخصصة للموقع المركزي والأسبوعين اللاحقين في المواقع الخارجية. ويعتمد تحقيق الأهداف المرجوة من عمليات النقل على حالة العضو الذي تم نقله وعلى طبيعة العمل، ولكنّا نعتبر أن نقل العائلات لمدة ستة شهور أو أكثر هو من الممارسات الأفضل في هذا المشروع.

كما في مشروع الأستديو العالمي، قمنا بعمل تحليل للعلاقات الاجتماعية (SNA) (انظر الفصل الثالث عشر من هذا الكتاب) في هذا المشروع. وظهرت تحليلات العلاقات الاجتماعية في أنماط التواصل بين أعضاء الفريق في جميع المواقع الخاصة بالمشروع. أما البيانات التي يتم إدخالها إلى تطبيق تحليل العلاقات الاجتماعية هي رسوم بيانية للشركة تتضمن اسم وموقع كل عضو من أعضاء الفريق. وقد تم إجراء مسح عن طريق شبكة الإنترنت لكل عضو من أعضاء الفريق لوصف علاقته بفريق المشروع. وتحتاج عملية المسح إلى عشر دقائق تقريباً لكل عضو من الأعضاء حتى تكتمل، لذلك يُعتبر هذا الأسلوب في جمع البيانات من الأساليب المريحة. تكون نتيجة هذا التحليل رسومات بيانية تشابكية تظهر روابط التواصل بين أعضاء الفريق. وعلى الرغم من أن تطبيق تحليل العلاقات الاجتماعية على نظام إدارة المباني الآلي لم يُظهر اختلافاً كبيراً مع المشاريع الموزَّعة الأخرى، فقد أصبح من الواضح أن وجود عمليات تواصل إضافية أكثر فاعلية بين أعضاء الفريق الواحد وأعضاء الفرق المختلفة سوف يعود بالفائدة على المشروع. وكانت إحدى الملاحظات المهمة أن نصف الأعضاء المعينين في المشروع والذين يظهرون في الرسم البياني المتعلق بشركة المشروع كانوا يعملون بدوام كامل في المشروع فقط. وكون أعضاء الفريق يعملون في المواقع التي تعمل عن بُعد بدوام جزئي يخلق صعوبات لمدير المشروع الموجود في

الموقع المركزي في عملية التخطيط للعمل وتنفيذ المشروع وفقاً للخطة.

فكرة مفيدة:

احذر من الفرق الوهمية.

تتطلب أنشطة معينة خلال المرحل الأولية من تصميم الهيكلية إلى مقدار كبير من التواصل بين أعضاء الفريق لإنجاز العمل. عادة ما تكون هذه الفرق صغيرة لكنها مكوّنة من خبراء في المجالات المختلفة. لذلك، من المهم تبادل المعلومات بين أعضاء الفريق لتوحيد المهارات المتنوعة. غير أنه عندما لا تكون هذه الفرق من مؤسسة واحدة، تكون عمليات التواصل مطولة. وقد تتأثر إنتاجية أعضاء الفريق أو جودة المنتج سلبياً بذلك. إضافة إلى ذلك، يعتبر هؤلاء الخبراء مصدر معلومات قيم للمؤسسة، وعادة ما يمتلكون سنوات عديدة من الخبرة. عندما يُوجَد هؤلاء الخبراء في المواقع التي تعمل عن بُعد، سوف يتم مقاطعتهم عن طريق توجيه الأسئلة لهم من أعضاء يعملون على منتجات تمت في مراحل سابقة. لذلك، غالباً ما تكون الفرق الوهمية غير ملتزمة بإنجاز عمل مكثف في وقت معين خلال المراحل المبكرة من المشروع حيث يكون الخبراء متاحين للتواصل مع أعضاء الفريق.

17-7 الملخص

على الرغم من أن مشروع نظام إدارة المباني الآلي ما زال مستمراً، فمن الصعب تحديد احتمالات نجاحه أو فشله، ولكن يمكن ملاحظة عدد من الممارسات الجيدة المتعلقة بالتواصل بين

أعضاء الفرق الضخمة والموزَّعة بشكل كبير. ويجب تنفيذ بعض الممارسات التي تم اتخاذها في المراحل المبكرة مثل عملية تصميم الهيكلية في موقع واحد لتوفير عمليات تواصل مكثفة بين أعضاء الفريق. ويجب أن يتم تجزيء العمل بحيث يتم إنجاز كل جزء في وقت معين لتسريع اتخاذ القرارات ولإظهار التقدم في سير المشروع. وقد يؤدي عمل الأعضاء أصحاب الوظائف المهمة جزئياً إلى الإضرار بتقدم المشروع، خصوصاً إذا كانوا يوجَدون بعيداً عن الفريق المركزي.

المراجع

Books

Paulish, Daniel J. *Architecture-Centric Software Project Management*. Boston; MA: Addison-Wesley, 2002.

Conferences

Herbsleb, James D., Daniel J. Paulish and Matthew Bass. «Global Software Development at Siemens: Experience from Nine Projects.» *Proceedings of the 27th International Conference on Software Engineering*, St. Louis, MO., 2005, pp. 524-533.



القسم السادس

ملاحظات ختامية



الفصـــــل الثامن عشر

الاستنتاجات

إن تطوير البرمجيات عملية ذات طابع معقَّد. وقد بيَّنت الدراسات التي أجرتها مجموعة ستانديش على مدى عشر سنوات أن نسبة كبيرة من المشاريع البرمجية تفشل أو تتجاوز الموازنة المقررة أو الجدول الزمني أو لا تحقق الأهداف المنشودة (Johnson, 2000). إن عمليات تطوير البرمجيات الموزَّعة المراكز في عدة دول التي تختلف عن بعضها البعض في نظام التوقيت تزيد من تعقيد العمل.

نلخص في هذا الفصل المعوقات والقضايا الأخرى المتعلقة بمشاريع تطوير البرمجيات الموزَّعة المراكز. وقد تم تلخيص الأمور الأساسية التي تناولها هذا الكتاب لتوضيح الآلية التي يمكن من خلالها تطوير البرمجيات بنجاح باستخدام فرق عمل موزَّعة في مناطق جغرافية مختلفة. وننصح القرَّاء بأن يقوموا باستخلاص ووصف الممارسات الجيدة التي يرونها مناسبة، ويتم التشديد هنا على أهمية التجريب في مجال هندسة البرمجيات.

18-1 قضايا تتعلق بتطوير البرمجيات الموزَّعة المراكز

تدفع الأمور الاقتصادية المتعلقة بتطوير البرمجيات المنظمات التجارية إلى اتباع أسلوب الانتشار العالمي. إن المرتَّب المنخفض لمهندسي البرمجيات الموجودين في مناطق معينة من العالم هو من الأمور التي تم أخذها بعين الاعتبار في المشاريع الموزَّعة كأسلوب يُتّبع لطرح المنتج في السوق بتكلفة أقل أو لإنتاج المنتج وطرحه في السوق بأكبر سرعة ممكنة. وتعزز المستلزمات والتقنيات الضرورية للاتصالات كشبكة الإنترنت تعاون مهندسي البرمجيات في ما بينهم بوسائل إلكترونية. وكما ذكرنا في الفصل الأول من هذا الكتاب، تشمل بعض الأمور التي تحفز عمليات تطوير البرمجيات بشكل موزع ما يأتي (Carmel, 1999):

- محدودية القوى العاملة المدربة على التقنية اللازمة لبناء الأنظمة الحالية المعقدة.
- توفّر مهندسي برمجيات بكلفة أقل في دول كالهند والبرازيل والصين وأوروبا الشرقية.
- إمكانية إجراء عمليات البرمجة بشكل مستمر من دون توقف بوجود مهندسين يعملون في دول مختلفة في أنظمة توقيتها.
- وجود بنية تحتية وتقنيات مسبقة خاصة بالتواصل وعمليات التطوير البرمجي.

على الرغم من وجود أسباب تجارية قوية لتوزيع مراكز عمليات تطوير البرمجيات في عدة مواقع، يعمل لحساب مديري المشاريع البرمجية الآن أشخاص في دول أجنبية لم يلتقوا بهم قطّ، ولا يوجد لديهم الفرصة لإعادة النظر في جودة عملهم وتقويمه أو في تقدم عمليات التطوير. لقد انتهى الوقت الذي يمكن أن يقوم به مدير المشروع بالعمليات الإدارية عن طريق التجول بين الموظفين وحسب.

إضافة إلى ذلك، يقوم مديرو المشاريع البرمجية في الوقت الحالي بإدارة مشاريع أكبر من المشاريع التي اعتادوا على إدارتها نظراً إلى كون فرق العمل التي تعمل عن بُعد متوفّرة وذات تكلفة أقل. وما يزيد من تعقيد المشروع والمخاطر المتعلقة به للتوصل إلى إنتاج ناجح كون المشاريع أصبحت أكبر بوجود أعضاء أكثر عدداً، ووجود قائمة كبيرة من الوظائف التي يلزم برمجتها، ووقت قصير لازم لطرح المنتج في السوق.

هناك ثمة صعوبات محتملة قد تواجه مدير المشروع البرمجي الموزع المراكز، وهو ما تم الإشارة إليه في جميع أجزاء الكتاب (Mockus and Herbsleb, 2001). وتتضمن هذه الصعوبات:

- وجود ترابط بين البنود المتعلقة بالعمل. وبالتالي قد تظهر صعوبات في تنسيق العمل بين مواقع البرمجة الموزَّعة إذا لم يتم بناء الهيكلية للنظام بحيث تتوافق مع المهام الموكلة إلى المواقع التي تعمل عن بُعد والتي تكون عبارة عن وحدات مستقلة نسبيّا أو مجموعات عمل.

- أصبح التنسيق من الأمور المهمة بسبب عدم تجانس العمليات. على سبيل المثال، قد تؤدي الاختلافات في تحديد الاختبارات اللازمة للوحدات إلى وجود تداخلات وتعارض، وقد يُؤثِّر عدم الالتزام بالجدول الزمني في أحد المواقع في المواقع الأخرى. وقد لا يتم الإبلاغ عن هذه الأمور والمشاكل في وقت مبكر، أو قد يؤدي الاختلاف في التوقيت إلى تسليم العمل من موقع لآخر بشكل متكرر نتيجة لعمليات التطوير المستمرة طوال الوقت.

- يكون بناء نموذج للمنتج الذي سيتم تطويره أكثر صعوبة في المواقع المتعددة، ما يصعب من قدرة الجميع على فهمه وإدراك كنهه. وهذا بلا شك يحتاج إلى وقت أطول للتواصل وإيجاد

صيغة عامة يدركها الجميع، إذ إن أفراد الفرق التي تعمل عن بُعد يدركون طبيعة المنتج بشكل أفضل من خلال البريد الإلكتروني والمناقشات الهاتفية. إن احتمالية الوصول إلى مسؤول الهيكلية في موقع عمله وطرح بعض الأسئلة عليه وطلب عرض تمثيلي للمنتج هي احتمالية محدودة للمبرمجين الذين يعملون عن بُعد، لذلك من الصعب تكوين رؤية مشتركة للجميع.

- يوجد اختلاف في البنية التحتية بين مواقع البرمجة المختلفة، يشمل طريقة ربط الشبكات ووسائل وأساليب تطوير البرمجيات والمختبرات الخاصة بالبرمجة والاختبار واختلاف إصدارات العمليات والأدوات الخاصة بالتعديل.

غالباً ما تكون أكبر المشاكل التي تواجه المشروع مرتبطة بعمليات التواصل بين المواقع لأن المشاركين في المشروع ذوو مستويات تقنية وثقافات مختلفة ومتنوعة ومعرفة مسبقة مختلفة في مجال العمل المطلوب. فمن غير المحتمل أن يكونوا قد شاركوا معاً في مشروع واحد مسبقاً؛ كما إنهم يملكون الخبرة في مجالات برمجية مختلفة ويتحدثون بلغات مختلفة. إضافة إلى ذلك، فمن غير المحتمل أن لا يكون المشاركون قد خططوا لاستخدام وسائل تواصل مع المواقع الأخرى نتيجة لعدم القدرة على الحديث معهم وجهاً لوجه.

18-2 وصفة للنجاح

استعرضنا في هذا الكتاب عدداً من العوامل المهمة للوصول إلى النجاح نعتقد بأنها تمثِّل العوامل الأساسية لنجاح مشاريع تطوير البرمجيات الموزَّعة المراكز. ففي حين أن معظم هذه العوامل مهمة أيضاً في المشاريع المنتظمة غير الموزَّعة، غير أنها مهمة بشكل أكبر

في مشاريع تطوير البرمجيات الموزَّعة المراكز. ويجب أن يُنظر إلى عملية تطبيق هذه العوامل كاستثمار في مجال الجودة وتقليل أثر المخاطر المحتملة. وعلى كل حال، يجب أن ندرك بأن عوائد هذا الاستثمار تكون أكبر بكثير بوجود علاقة لفترة زمنية طويلة مع المزود.

نشير إلى هذه العوامل المهمة للنجاح بشكل عام كما يأتي:

- **الحد من الغموض**: إن عملية تطوير البرمجيات ذات طبيعة غير واضحة يكتنفها الغموض. ويتم التعامل مع وجود الكثير من عدم الوضوح والالتباس في المشاريع البرمجية عن طريق التواصل غير الرسمي، والتي تتم بشكل أسهل إذا كانت الفرق المتعاونة غير موزَّعة. وتتطلب مشاريع تطوير البرمجيات الموزَّعة المراكز الكثير من الأفكار والعمل لتكوين توافق حول الطريقة التي سوف تعمل وتتواصل بها الفرق التي تعمل عن بُعد بحيث يحصل أفراد الفريق على المعنى المقصود لمهام العمل المتعلقة بالمشروع.
- **الوصول إلى أقصى حد من الثبات**: إن لعدم الثبات في مجالات المشروع المختلفة تأثيراً سلبياً كما لتأثير عدم الوضوح. ويعتبر التغير في الطلبات، وبالأخص التي تأتي في وقت متأخر، خطيرة ويجب مراقبتها والتعامل معها بحذر. كما يمكنها أن تؤدي إلى إبطاء العمل في مشاريع تطوير البرمجيات الموزَّعة المراكز بشكل كبير؛ لذلك، نحن ننصح بأن يتم إنشاء مهام ثابتة للمشروع بشكل متأنٍّ مثلما يتم في عملية تحديد المتطلبات وإنشاء الهيكلية، حتى وإن تطلب ذلك تأخيراً في البدء في مرحلة الإنشاء للمنتج.
- **فهم العلاقات الترابطية**: تحدد طبيعة الترابط الموجود بين المهام الموكلة للفرق التي تعمل عن بُعد مدى التنسيق اللازم بين هذه الفرق.

- لذلك، من المهم جداً فهم هذه العلاقات الترابطية للتخطيط للمشروع وتنفيذه. على سبيل المثال، إذا كان هناك مهمتان مترابطتان بشكل كبير، يفضل نقلهما إلى فريق واحد أو فريقين قريبين من بعضهما بعضاً، مع الأخذ بعين الاعتبار ضرورة أن تتقاطع فترات العمل للفريقين، وأن يكون الفريقان، كذلك، قد عملا مع بعضهما البعض من قبل.
- **تسهيل التنسيق**: عندما تعمل الفرق على إنجاز مهام مترابطة، تبرز الحاجة إلى التنسيق في ما بينها. لذلك، يجب أن يتناسب مستوى التنسيق مع احتياجات العمل. والتواصل هو إحدى وسائل التنسيق، ولكن يمكن أن يتم التنسيق بين الفرق في العمليات والممارسات الإدارية وهيكلية المنتج، على سبيل المثال لا الحصر. ويجب الموازنة بين اختيار الجهود التي تبذل للتنسيق بين هذه الأمور والمخاطر التي تنشأ في حال عدم التنسيق بينها. ويجب عدم الإهمال وبذل الجهود اللازمة في تحسين العملية والعمليات التلقائية ووجود أدلة مساعدة للعمليات البرمجية.
- **الموازنة بين المرونة والصلابة**: تختلف الفرق التي تعمل عن بُعد في معرفتها والمعلومات التي تمتلكها عن مجال العمل واللغة والثقافة والممارسات المستخدمة داخل الشركة. يجب أن تكون العمليات البرمجية مرنة بما يكفي للتوفيق بين هذه الاختلافات. وفي نفس الوقت، يجب أن تكون بعض المجالات في العمليات البرمجية محددة بشكل واضح وجيد لأنها تُوفِّر لمدير المشروع المعلومات حول سير عمل الفرق التي تعمل عن بُعد في غياب ميزة القدرة على تتبع سير العمل عن طريق التجوال بين الفرق.

لقد قمنا بتحديد نظام سير للعمل يرتكز على مبدأ المراحل المتتالية بحيث يتم تطبيق أفضل الممارسات التي يتم تدوينها عن

عوامل النجاح المهمة والحرجة. وتُعتبر الأنشطة التي تُنجز في مراحل مبكرة مثل تطوير نموذج للمتطلبات ووصف الهيكلية مهمةً وحرجةً لتوفير فهم للعمل وتقسيمه إلى حزم يتم توزيعها على الفرق التي تعمل عن بُعد. ويتم تنفيذ أنشطة المراحل المبكرة عن طريق فرق صغيرة ومتجمعة تضم الأعضاء المميزين من المواقع المشاركة تعمل عن بُعد. ومن دون وجود توثيق جيد لمتطلبات المنتج وهيكلته، فإننا لا نتوقع أن تكون الفرق التي تعمل عن بُعد قادرة على إنجاز المنتج بنجاح.

كما تشير أبحاثنا وخبراتنا السابقة إلى وجود مهام يمكن توزيعها على عدة مواقع، وأخرى من الأفضل أن يتم إنجازها في مجموعة منتظمة غير موزَّعة. على سبيل المثال، الأنشطة التي تتم في المراحل المبكرة، مثل تصميم الهيكلية، تحتاج إلى قدر كبير من التفاعل بين أعضاء الفريق وينبغي ألا يتم توزيعها على عدة مواقع. وبعد أن يتم تحديد هيكلية النظام وتحديد حزم العمل التي تكون نسبياً غير مرتبطة مع بعضها بعضاً، يمكن أن تبدأ الفرق البرمجية الموزَّعة العمل على هذه المهام.

إن عملية تجزيء العمل المعقد إلى مراحل متتالية بحيث تنجز كل مرحلة شهرياً من شأنها أن تساعد مدير المشروع الموزع في الحصول على رؤية أفضل وأكثر وضوحاً للتقدم في العمل وتساعد في تركيز الفرق الموزَّعة على تحقيق أهداف شهرية منسقة. ويتم اختبار النتائج البرمجية لشهر واحد خلال الشهر التالي والوقت الذي تستمر فيه الفرق البرمجية في البلورة والتحويل المستمر والمتزايد للمقاطع العمودية للمنتج إلى وظائف عمل.

ننصح بإبقاء عدد الأعضاء في المجموعة الواحدة أقل من عشرة أشخاص، إذ إن هذا هو العدد الأفضل لتنفيذ عمليات برمجية

سريعة. ويتطلب ذلك أن يكون عدد المصممين ضمن قيود معينة بحيث يتناسب مع حجم العنصر الواحد في الهيكلية، بحيث يمكن لفريق البرمجة الذي يعمل عن بُعد أن ينفِّذ أحد المكوّنات من خلال عشرة أعضاء وبزمن يقل عن سنة واحدة.

تزيد عمليات التواصل في المشاريع الكبيرة والمعقدة والموزَّعة المراكز بشكل سريع، بحيث تصبح خارجة عن السيطرة ما لم تتم المساعدة في وضع القواعد التي تنظِّم طريقة تواصل الفرق مع بعضها، وكذلك إنشاء بنية تحتية تدعم هذا. على سبيل المثال، ننصح بتقييد استخدام البريد الإلكتروني في إيصال المعلومات ذات الطبيعة السريعة والعابرة؛ ولكن، لأسباب تتعلق بتعقد الأمور المتعلقة بالمعلومات المفيدة للمشروع والتي قد تلزم المثابرة لفترة طويلة، يعضِّل استخدام منتديات المناقشة. ويجب استثمار الكثير من الجهد الذهني في إنشاء استراتيجية سليمة لإدارة المعرفة. ويمكن استخدام الأدوات التعاونية مثل نظام Wiki لإدارة محتويات هذه المنتديات.

تصبح الفرق الموزَّعة أكثر إنتاجية عندما تعمل معاً لفترة من الوقت. لذلك، يجب اعتبار استراتيجية نقل العمل إلى الخارج كاستراتيجية طويلة الأمد بحيث يتم تطوير موقع المبرمجين البعيد عن المركز الرئيس ليصبح مركزاً مختصاً ذا خبرة. ولا ننصح بأن يكون تقليل الكلفة على المدى القصير الهدف الوحيد من التطوير الموزع. وعلى الرغم من إمكانية تحقيق هذا الهدف غير أن خبرتنا في هذه المشاريع لم تؤكد توفيراً في التكاليف على المدى القصير.

على الرغم من الطريقة الممتازة التي وضَّح بها الفريق المركزي المواصفات، غير أننا نعتقد أنه ما زال من الصعب على أعضاء الفرق التي تعمل عن بُعد أن يفهموا كل ما يقرأونه، خصوصاً إذا لم تكن

لغة المشروع هي لغتهم الأم. إضافة إلى ذلك، حتى المشاريع المخطط لها بشكل جيد تكون عرضة للتغيير مع استمرار سير العمل ومواصلة إيجاد حلول لأي أمور طارئة، وتزداد صعوبة العملية حين العمل مع أجانب. لذلك، ننصح بالسفر لأهميته في المراحل المبكرة للمشروع، والتواصل وجهاً لوجه، وتبادل أعضاء فرق العمل لفترات طويلة في ما بين المواقع الموزَّعة.

18-3 تبادل أفضل المنهجيات

يُشجع العمل في المشاريع على مشاركة أفضل المنهجيات في تطوير البرمجيات الموزَّعة المراكز، كما ورد في فصول هذا الكتاب. ونتيجة للكم القليل المتوفّر للعامة حول تطوير البرمجيات الموزَّعة المراكز، نشعر أننا حددنا أفضل المنهجيات التي يمكن تطبيقها على المشاريع المستقبلية، على الأقل بالنسبة إلى مشروع من عدة مشاريع موثقة. وتتمثل المشكلة الأساسية في أن كل مشروع يُعتبر فريداً من نوعه ومختلفاً عن المشاريع الأخرى بشكل كبير من ناحية منهجية تطوير البرمجيات والمجالات والحجم والعمليات المستخدمة. لذا، لا يوجد ضمانات أن المنهجيات التي تم استخدامها في أحد المشاريع بنجاح سوف تنجح حتى ولو بشكل جزئي إذا طُبقت في مشاريع أخرى. ومع ذلك، نحن لا نعتقد أنه يمكن تحسين انضباط مهندسي البرمجيات دون وجود مشاركة واسعة لأفضل المنهجيات والتجارب السابقة. يتم استخدام كتيب خاص بأفضل المنهجيات والتجارب السابقة والطبقات التنظيمية والأنماط المضادة لإضافة خبرات الآخرين إلى خبرات مهندسي البرمجيات ليقوموا بالتالي بالتخطيط الجيد لنجاح المشروع.

من الطرق الأخرى لاختبار صلاحية المنهجيات المتَّبعة في هندسة البرمجيات «التجريب». فقد كنا محظوظين لتطبيق ذلك في

كل من جامعة كارنيغي ميلون، وجامعة هارفارد التجارية، والمعهد الدولي لتكنولوجيا المعلومات في بانغالور، وجامعة مونموث، وجامعة ولاية بنسلفانيا، وجامعة ميونخ التقنية، وجامعة لومنيتش، والجامعة البابوية الكاثوليكية في ريو غراند دو سول. وأدت هذه المشاركات إلى تطوير مشروع الأستديو العالمي على مدى عدة سنوات؛ وكان المشروع كبيراً بحيث تمت الاستعانة بفريق من المبرمجين من طلاب الجامعات من خمس دول في أربع قارات، بحيث أمكن دراسة ثقافات وتنظيمات وأساليب تقنية مختلفة. وعلى الرغم من أن استخدام فرق من الطلاب لا يناسب البيئات الصناعية تماماً، غير أن التجربة كانت عبارة عن عملية محاكاة مفيدة. على سبيل المثال، يحصل الطلاب الخريجون على درجات إضافية عن طريق مشاركتهم في المشروع كجزء من متطلب التدريب العملي بدلاً من حصولهم على عائد مادي.

على الرغم من ذلك، هناك احتمال أن يكافأوا على عملهم الجيد. كذلك، يمتلك العديد من الطلاب الحاليين الخريجين في هندسة البرمجيات خبرة صناعية جيدة؛ لذا، لا تختلف خبراتهم ومهاراتهم كثيراً عن مهارات المهندسين الذين يتم تزويدهم من شركات خارجية نظراً إلى تكاليف أعمالهم المنخفضة.

إن فائدة هذا المشروع التجريبي الكبرى هي عدم وجود مخاطر تجارية قد تنتج من فشل المشروع. لذا، يستطيع فريق الأبحاث الاستفادة من حالتي النجاح والفشل. كما يمكن تطبيق التقنيات الواعدة والأدوات والعمليات بشكل تجريبي، بحيث تكون نتيجة ذلك تقليل المخاطر وليس الدخول في المخاطر المترتبة على تجريب أساليب جديدة. فليس ثمة مدير مشروع برمجي يرغب في أن يكون مشروعه مجالاً للتجربة.

تم تجهيز هذا المشروع بحيث يتم جمع أكبر قدر ممكن من البيانات النوعية في كل مرحلة من مراحله. وقد أصبحت هذه البيانات متوفّرة للباحثين في مجال هندسة البرمجيات. وبهذه الطريقة، يمكن أن نكوّن صورة أفضل عمّا يمكن أن يكون مفيداً وما قد يكون غير مفيد في مجال تطوير البرمجيات الموزَّعة المراكز.

18-4 الملخص والاستنتاجات

أصبح من السهل الإفادة من هذا العالم باتساعه (Friedman, 2005)، خصوصاً بالنسبة إلى هندسة البرمجيات. وليس في نيتنا أن نقوم بتحليل الأمور الاقتصادية والسياسية المتعلقة بتطوير البرمجيات الموزَّعة المراكز. وهناك ثمة دليل قوي على أن العولمة هي توجّه لا يمكن التصدي له؛ ونتيجة لذلك، أصبح التوجه نحو مشاريع تطوير البرمجيات الموزَّعة المراكز أمراً حتمياً. لذلك، نهدف إلى فهم الآلية التي يمكن من خلالها تحقيق نتائج جيدة في مشاريع تطوير البرمجيات الموزَّعة المراكز. ونعتقد أنه يمكن الإفادة من أفضل الممارسات والتجارب التي تم اقتراحها في المشاريع الضخمة والمعقدة والمنتظمة (غير الموزَّعة). فإذا كنت مهندس برمجيات، يجب أن تدرك أن عالم الأعمال التجارية في هذه الأيام يزيد من فرص العمل والتعاون مع زملاء في دول أخرى. ويجب أن تكون مستعداً لذلك، ونأمل أن تمتلك الخبرة التي تؤهلك للعمل مع مهندسين أجانب. ونأمل أن تستمتع في العمل بهذه الطريقة وبحياتك المتعلقة به، وأن تجد الوقت الكافي لتتمتع بحياتك بينما تقوم بتكوين صداقات مع مهندسين من دول أخرى.

نأمل أن تجد بعض الاقتراحات والنصائح المفيدة والممارسات التي دار الحديث حولها في مجال مشاريع تطوير البرمجيات الموزَّعة

المراكز. وإن أفضل مكافأة يمكن أن يحصل عليها أي مهندس برمجيات هي رؤية المنتج الذي قام ببرمجته قيد الاستخدام. وستتاح للمهندسين الذين يعملون في المشاريع الموزَّعة الفرصة للسفر حول العالم، وسوف يتعلمون العمل مع أشخاص ذوي ثقافات وخلفيات مختلفة. ونعتقد بأنك ستستمتع بالعمل في مثل هذه المشاريع الموزَّعة عندما تحاول التعامل مع التعقيدات والتغلب على المخاطر المتعلقة بإنشاء منتج ناجح. ونتمنى لك حظاً سعيداً في مشروعك القادم في تطوير البرمجيات الموزَّعة المراكز.

المراجع

Books

Carmel, Erran. *Global Software Teams: Collaborating Across Borders and Time Zones*. Upper Saddle River, N. J.: Prentice Hall, 1999.

Friedman, Thomas. *The World is Flat: A Brief History of the Twenty-First Century*. [n. p.]: Farrar, Straus and Giroux, 2005.

Periodicals

Johnson, Jim. «Turning Chaos into Success.» *Software Magazine*: vol. 19, no. 3, December 1999/ January 2000, pp. 30-39.

Conferences

Mockus, A. and James D. Herbsleb. «Challenges of Global Software Development.» *Proceedings of the Seventh International Software Metrics Symposium,* London, England, 4-6 April 2001, pp. 182-184.

الثبت التعريفي

إصدار هندسي (Engineering Release): تسليم تراكمي لحزمة العمل في نهاية كل فترة شهرية من فترات البرمجة التكرارية؛ حيث يتم في نهاية كل فترة برمجية تكرارية تجميع الإصدارات الهندسية التي تشكل جزءاً من المنتج القابل للتنفيذ.

تنسيق (Coordination): إدارة الملحقات والروابط التي تتم أثناء العمل على إنجاز الأنشطة المختلفة.

جدول زمني (Schedule): خطة لتنفيذ مهام البرمجة والتطوير بحيث تحدد فيها الأزمنة المحددة لتنفيذ تلك المهام.

خطة الإصدار (Release Plan): وصف تسويقي رفيع المستوى لإصدارات المنتج العديدة المزمع تسويقها وبيعها.

خطة البرمجة والتطوير (Development Plan): خطة تتضمن تقديراً للجدول الزمني للبرمجة وتحدد الموارد البشرية اللازمة وتكاليف التنفيذ.

خطة التكامل والاختبار (Integration and Test Plan): خطة

تعرض كيفية دمج وتكامل خصائص ومزايا المنتج والاختبار اللازم بناءً على ترتيب إصدارها.

خطة المشروع (Project Plan): مستند يتضمن خصائص الإصدار وخطة البرمجة والتطوير وخطة الاختبار والتكامل وخطة الإصدار.

عنصر/ مكوّن (Component): كيان ذو قابلية توزيع وانتشار بشكل مستقل أثناء وقت التشغيل.

فترة برمجية تكرارية (Iteration): فترة تُحدد عدداً من الفترات الشهرية السريعة والتي تؤدي بمجموعها إلى الإصدار القابل للتنفيذ لجزء من المنتج

فترة زمنية شهرية (Sprint): فترة مدتها شهر ضمن مدة الفترة الزمنية البرمجية، وينتج من هذه الفترة إصدار هندسي في مرحلة التنفيذ.

قدرة على التنسيق (Coordination Capability): قدرة الشركة أو المنظمة على إدارة الروابط التي تتم أثناء العمل على إنجاز الأنشطة المختلفة.

مدير التزويد (Supplier Manager): إحدى الوظائف في المركز الرئيس، يقوم مدير التزويد بالمساعدة في إدارة العلاقات والمشاريع مع المواقع التي تعمل عن بعد أو مع مزودي الوحدات البرمجية التي يجب تطويرها.

مدير المشروع (Project Manager): شخص مسؤول عن إعداد الخطة لدورة حياة المنتج كاملةً، ويقوم في المقام الأول بإعداد خطط المشروع لتنفيذ المنتج المطلوب وتنفيذ تحليل المخاطر

وعرض الاستراتيجيات الخاصية للتخفيف من حدة وتأثير المخاطر.

مدير المنتج (Product Manager): شخص مسؤول عن المنتج في المقام الأول، ويعمل على تحديد احتياجات السوق في ما يتعلق بالمنتج، ويعمل على إعداد خطط الإصدار وذلك بتعريف وتحديد حزم الخصائص والمزايا التي تشكل حلولاً برمجية يمكن بيعها.

مواصفات إصدار مزايا وخصائص النظام (Feature Release Specification): تجميع الخصائص وإضافتها إلى الإصدارات التي يمكن بيعها.

وحدة (Module): وحدة تنفيذ برمجية ثابتة.

CMM® و**CMMI**: هما علامتان تجاريتان لمكتب البراءات والعلامات التجارية في الولايات المتحدة المسؤولة عنه جامعة كارنيغي ميلون.

RUP®: هو علامة تجارية مسجلة لشركة IBM.

SMATAM: طريقة تحليل تبادل الهيكلية وSEPG هما علامتا خدمات خاصة بجامعة كارنيغي ميلون.

CORBA™و**UML**: هما علامتان تجاريتان لشركة Object Management Group في الولايات المتحدة الأميركية و/ أو دول أخرى.

J2EE™: هو علامة تجارية مسجلة لشركة صن مايكروسيستمز في الولايات المتحدة الأميركية و/ أو دول أخرى.



ثبت المصطلحات

آلاف السطور من شيفرة البرمجة KLOC: Thousand Lines of Code

أجهزة القياس والتحكم، أو الصناعي والتجاري I&C: Instrumentation & Control or Industrial & Commercial

إدارة إعداد البرمجيات SCM: Software Configuration Management

إدارة التهيئة CM: Configuration Management

إدارة الجودة QM: Quality Management

إدارة الجودة الشاملة TQM: Total Quality Management

إدارة الغذاء والدواء الأميركية FDA: Food & Drug Administration

إدارة بالأهداف MBO: Management by Objectives

إدارة بالتجول MBWA: Manage by Walking Around

أسلوب تحليل للمفاضلة الهيكلية ATAMSM: Architecture Tradeoff Analysis Method

إصدار R: Release

إصدار هندسي	ER: Engineering Release
إصدار خاص للاختبار	TR: Test Release
أمر تغيير هندسي	ECO: Engineering Change Order
أهداف/ أسئلة/ قياسات	G/ Q/ M: Goals/ Questions/ Metrics
بأسرع وقت ممكن	ASAP: As Soon As Possible
بحث وتطوير	R&D: Research & Development
برمجيات	SW: Software
برمجيات جاهزة تجارية	COTS: Commercial Off-the-Shelf
برنامج Java النسخة 2، المخصصة للشركات التجارية	J2EE™: Java 2 Platform, Enterprise Edition
البرنامج الاستراتيجي الأوروبي للبحث والتطوير (في مجال تكنولوجيا المعلومات)	ESPRIT: European Strategic Programme for Research and Development
برنامج تحفيز طويل الأمد	LTIP: Long-Term Incentive Program
بروتوكول نقل النص التشعبي	HTTP: Hypertext Transfer Protocol
بروتوكول الوصول المفتوح البسيط	SOAP: Simple Open Access Protocol
بيئة التطوير المتكامل	IDE: Integrated Development Environment
تبادل الفرع الخاص	PBX: Private Branch Exchange
تحقق ومصادقة	V&V: Verification & Validation
تحليل باستخدام نظام احتساب النقاط	FPA: Function Point Analysis
تحليل شامل	GA: Global Analysis

تحليل الشبكة الاجتماعية SNA: Social Network Analysis

تخطيط مشاريع البرمجيات ذات الهيكلية المركزية ACSPP: Architecture-Centric Software Project Planning

تخطيط وتنظيم وتنفيذ وقياس POIM: Planning, Organizing, Implementing, Measuring

تخمين بري دفين علمي أو تخمين بري دفين الساذج SWAG: Scientific Wild-Assed Guess or Silly Wild-Assed Guess

تدفئة وتهوئة وتكييف الهواء HVAC: Heating, Ventilation, & Air Conditioning

تصميم محكوم بالخصائص ADD: Attribute Driven Design

تصميم رفيع المستوى HLD: High-Level Design

تطوير البرمجيات الموزَّعة المراكز GSD: Global Software Development

تغيّر القيمة COV: Change-of-Value

تقرير عن مشكلة PR: Problem Report

تقرير عن مشكلة معروفة KPR: Known Problem Report

تقرير مشاكل البرمجية SPR: Software Problem Report

تقنية تقويم ومراجعة البرنامج PERT: Program Evaluation & Review Technique

تقويم مخاطر البرمجية SRE: Software Risk Evaluation

تكامل نموذج نضج القدرة CMMI: CMM Integration

تكنولوجيا المعلومات IT: Information Technology

تنفيذ الطريقة عن بُعد RMI: Remote Method Invocation

CDC: Connected Device Configuration تهيئة جهاز موصول

IPC: Inter-process Communication تواصل بين العمليات

PUCRS: Pontifical Catholic University of Rio Grande de Sul الجامعة البابوية الكاثوليكية في ريو غراندي دو سول

CMU: Carnegie Mellon University جامعة كارنيغي ميلون

UL: University of Limerick جامعة ليميريك

MU: Monmouth University جامعة مونموث

TUM: Technical University of Munich جامعة ميونيخ التقنية

BCS: British Computer Society الجمعية البريطانية للحاسوب

CSA: Computing Services Association or Canadian Standards Association جمعية الخدمات الحاسوبية أو جمعية المقاييس الكندية

ACM: Association of Computing Machinery جمعية المعدات الحاسبة

PC: Personal Computer حاسوب شخصي

SDP: Software Development Plan خطة تطوير البرمجيات

SDLC: Software Development Life Cycle دورة حياة تطوير البرمجيات

PROM: Programmable Read-Only Memory ذاكرة القراءة فقط القابلة للبرمجة

EEPROM: Electrically Erasable Programmable Read-Only Memory ذاكرة القراءة فقط القابلة للمسح والبرمجة كهربائياً

RAM: Random Access Memory ذاكرة الوصول العشوائي

MTTR: Mean-Time-To-Repair زمن وسطي إلى حالة إصلاح

MTTF: Mean-Time-To-Failure	زمن وسطي إلى حالة فشل
MTBF: Mean-Time-Between-Failures	زمن وسطي بين حالات الفشل
LOC: Lines of Code	سطور من شيفرة البرمجة
WWW: World Wide Web	شبكة الإنترنت العالمية
ISDN: Integrated Services Data Network	شبكة بيانات الخدمات التكاملية
IHN: Integrated Health Network	شبكة الصحة التكاملية
LAN: Local Area Network	شبكة محلية
QA: Quality Assurance	ضمان الجودة
MSLite: Management Station Light	ضوء محطة الإدارة
PMW: Project Manager Workbench	طاولة عمل مدير المشروع
CPM: Critical Path Method	طريقة (تحديد) المسار الحرج
MR: Modification Request	طلب إجراء تعديلات
RPC: Remote Procedure Call	طلب إجراء عن بُعد
ECR: Engineering Change Request	طلب تغيير هندسي
NLOC: Net Lines of Code	عدد أسطر شيفرة البرمجة الصافية
DLOC: Delta Lines of Code	عدد سطور شيفرة البرمجة (المضافة أو المعدلة)
PPP: Product Planning Process	عملية تخطيط المنتج
RUP®: Rational Unified Process	عملية موحدة منطقية
C&C: Component & Connector	عنصر ورابط

URL: Universal Record Locator	عنوان الموقع الإلكتروني على الإنترنت
DB: Data Base	قاعدة البيانات
CD: Compact Disc	قرص مضغوط
ERB: Engineering Review Board	لجنة المراجعة الهندسية
XML: eXtensible Markup Language	لغة الترميز الممتدة
UML™: Unified Modeling Language	لغة النمذجة الموحدة
MS: Microsoft	مايكروسوفت
PLC: Programmable Logic Controller or Power Line Communications	متحكم منطقي قابل للبرمجة أو اتصالات خط الطاقة
ASR: Architecturally Significant Requirement	متطلب مهم من الناحية الهيكلية
OMG: Object Management Group	مجموعة إدارة الكائنات
SEPGSM: Software Engineering Process Group	مجموعة عمليات هندسة البرمجيات
PCG: Product Control Group	مجموعة مراقبة المنتج
FSS: Field System Simulator	محاكي نظام حقل العمل
SEL: Software Engineering Laboratory	مختبر هندسة البرمجيات
MTS: Microsoft Transaction Manager	مدير معاملات مايكروسوفت
CDR: Critical Design Review	مراجعة نقدية للتصميم
ASP: Application Service Provider	مزود خدمة برامج تطبيقية
GSP: Global Studio Project	مشروع الأستديو العالمي

مصفوفة هيكلية التصميم	DSM: Design Structure Matrix
معالج الإشارة الرقمية	DSP: Digital Signal Processor
معالجة إلكترونية للبيانات	EDP: Electronic Data Processing
معدات	HW: Hardware
معهد أبحاث شركة سيمنز	SCR: Siemens Corporate Research, Inc.
معهد هندسة البرمجيات	SEI: Software Engineering Institute
المعهد الوطني الأميركي للمعايير	ANSI: American National Standards Institute
مكتبات الربط الديناميكية	DLL: Dynamic Link Library
ملايين السطور من شيفرة البرمجة	MLOC: Million Lines of Code
منطقة العمليات الأساسية	KPA: Key Process Area
منظمة المقاييس والمواصفات الدولية	ISO: International Standards Organization
مواصفات إصدار الخصائص	FRS: Feature Release Specification
مواصفات إصدار المكوّنات	CRS: Component Release Specification
مواصفات متطلبات التسويق	MRS: Marketing Requirements Specification
مواصفات متطلبات النظام	SRS: System Requirements Specification
موزع القيمة المضافة	VAR: Value-Added Reseller
نائب الرئيس	VP: Vice President
نشر وتوزيع وظيفة الجودة	QFD: Quality Function Deployment

نظام أتمتة المباني	BAS: Building Automation System
نظام احتساب النقاط (للبرامج)	FP: Function Points
نظام إدارة قاعدة البيانات	DBMS: Data Base Management System
نظام التحكم بالشيفرة المصدرية	SCCS: Source Code Control System
نظام التحكم بالمراجعة	RCS: Revision Control System
نظام الترميز الأميركي لتبادل المعلومات	ASCII: American Standard Code for Information Exchange
نظام مالي	FS: Financial System
نظام مراقبة البرنامج	PCS: Project Control System
نظام معالجة البيانات	DPS: Data Processing System
نظم المعلومات الصحية	HIS: Health Information System
نمذجة الكائن المضمن ذو الزمن الحقيقي	ROOM: Real-time Object-Oriented Modeling
نموذج إزالة العيوب	DRM: Defect Removal Model
نموذج لتواصل البرمجيات الموزَّعة (على شبكة)	DCOM: Distributed Component Object Model
نموذج لحساب تكلفة البرمجيات	COCOMO: Constructive Cost Model
نموذج نضج القدرة	CMM: Capability Maturity Model
هندسة البرمجيات بمساعدة الحاسوب	CASE: Computer-Aided Software Engineering
هندسة برمجيات الجودة	SQE: Software Quality Engineering

هندسة المتطلبات	RE: Requirements Engineering
هيكلية خط الإنتاج	PLA: Product Line Architecture
هيكلية وسيط لتسهيل طلبات بين برمجيات (مختلفة اللغة)	CORBA™: Common Object Request Broker Architecture
هيئة مهندسي الإلكترونيات والكهرباء	IEEE: Institute of Electrical and Electronics Engineers
واجهة برمجة التطبيقات	API: Application Programming Interface
واجهة المستخدم	UI: User Interface
واجهة المستخدم الرسومية	GUI: Graphical User Interface
وثيقة التصميم التفصيلية	DDD: Detailed Design Document
وثيقة التصميم رفيع المستوى	HLDD: High-Level Design Document
وحدة المعالجة المركزية	CPU: Central Processing Unit
ورشة عمل حول سمات الجودة	QAW: Quality Attribute Workshop
وصول مباشر للذاكرة	DMA: Direct Memory Access
الوطنية للملاحة الجوية والفضائية (ناسا)	NASA: National Aeronautics and Space Administration



الفهرس

ـ ت ـ

- ج -

- خ -

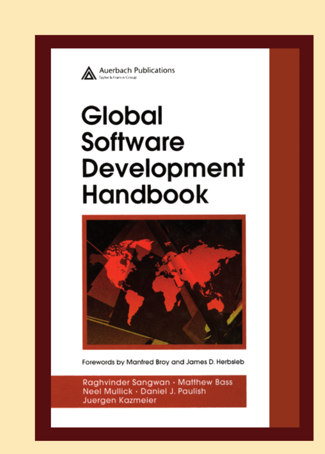
ـ ف ـ

ـ ن ـ

ـ هـ ـ

ـ و ـ

دلـيـل تطوير البرمجيّات الشامل(*)



(*) الكتاب الأول من تقنية المعلومات

السلسلة: تضم هذه السلسلة ترجمة لأحدث الكتب عن التقنيات التي يحتاج إليها الوطن العربي في البحث والتطوير ونقل المعرفة إلى القارئ العربي.

الكتاب: يركّز الكتاب على التقنيات الملائمة لإدارة تطوير البرمجيّات والاتصالات الفاعلة، بالإضافة إلى التنسيق والتحكّم بتوزيع البيئات المُطوّرة للبرمجيّات، كما يشدّد على الاستراتيجيات المتعلّقة بمراقبةِ نوعيةِ البرامج الخاصة بتوزيع البرمجيّات المطوّرة وإنتاجها، ويرسي أسساً قوية لأفضل الممارسات وأكثرها عملية في هذا الحقل.

المؤلف: **راجفيندر سانغوان**: أستاذ متخصص بهندسة البرمجيّات - جامعة بنسلفانيا.

ماثيو باس: مهندس برمجيّات - جامعة كارنيغي ميلون.

نيل موليك: مهندس برمجيّات، ومتخصص بالتسويق الدولي - جامعة كارنيغي ميلون.

دانيال ج. باوليش: دكتوراه في الهندسة الكهربائية - معهد نيويورك للبوليتكنيك.

جيورغين كازمير: دكتوراه في الرياضيات وعلم الحاسوب - جامعة ميونيخ التكنولوجية.

المترجمة: **مرفت سلمان**: ماجستير في أنظمة إدارة المعلومات - جامعة عمّان العربية للدراسات العليا.

سلسلة كتب التقنيات الاستراتيجية والمتقدّمة

1. المياه
2. البترول والغاز
3. البتروكيمياء
4. النانو
5. التقنية الحيوية
6. تقنية المعلومات
7. الإلكترونيات والاتصالات والضوئيات
8. الفضاء والطيران
9. الطاقة
10. المواد المتقدمة
11. البيئة




الثمن: 25 دولاراً
أو ما يعادلها

ISBN 978-9953-0-1994-9
9 789953 019949

دليـل تطوير البرمجيّات الشامل

راجفيندر سانغوان - ماثيو باس - نيل موليك
دانيال ج. باوليش - جيورغين كازمير

(6 - 1)